2008-06-18

A Naive, Robust and Stable State Estimate

Todd Gordon Remund
Brigham Young University - Provo

A NAIVE, ROBUST, AND STABLE STATE ESTIMATE

by

Todd G. Remund

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Statistics

Brigham Young University

August 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Todd G. Remund

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	H. Dennis Tolley, Chair
Date	Scott D. Grimshaw
Date	John S. Lawson

ABSTRACT

A NAIVE, ROBUST, AND STABLE STATE ESTIMATE

Todd G. Remund

Department of Statistics

Master of Science

A naive approach to filtering for feedback control of dynamic systems that is robust and stable is proposed. Simulations are run on the filters presented to investigate the robustness properties of each filter. Each simulation with the comparison of the filters is carried out using the usual mean squared error. The filters to be included are the classic Kalman filter, Krein space Kalman, two adjustments to the Krein filter with input modeling and a second uncertainty parameter, a newly developed filter called the Naive filter, bias corrected Naive, exponentially weighted moving average (EWMA) Naive, and bias corrected EWMA Naive filter.

ACKNOWLEDGEMENTS

I give thanks to my wife Kristin and my children, (Dallin, Jason, Ethan, Kalli, and Julia), for putting up with a busy dad and husband. For always being there to lighten my mental load and loving me. Dr. H. Dennis Tolley, my advisor, for having an infinite amount of patience on my behalf. Despite the many snags and turns in my studies and research he has always been a wise and gracious mentor to me. He has supported me as a needy student and as a family man. My parents have given me loads of support, help, and love in getting me through. I thank the department for giving me the chance to get a Masters degree. I greatly appreciate all they have done.

CONTENTS

TABLES

FIGURES

1. INTRODUCTION

In the modern world, we see change in almost every field of study. In the arena of modeling and filtering, there is a move toward use of discrete filters rather than the continuous analog filter. This move introduces the use of the digital state space model or representation, where interest lies in estimating the state of a process. Migration toward discrete filters is inspired by the greater versatility that they offer in controlling processes.

In time series modeling, digital filters are used as tools for many applications. In finance they give the general trend of a stock or currency exchange. Doctors rely on filters to give them reliable readouts on an electrocardiogram, and patients trust in the technology that aids in the diagnosis of health problems. Airplanes process radar signals using digital filters to give safe passage through the atmosphere, and routine adjustments are made to typical flight to smooth out the bumps in the air. Fighter pilots find it easier to acquire enemy planes or drop a bomb on a pinpoint target through use of time series filters.

Filters are important in process control to help forecast the effects of the current measurements and any action taken to control the system. In fact, the filter is used directly to find the control action to stabilize a process. As discussed previously, many different disciplines use filters directly or indirectly. Much literature has been devoted to this topic with many facets of expression. Engineers look at the control situation differently than do mathematicians or statisticians, and there are many different ways of writing down the models and filters in matrix form or as transforms. The focus here is limited to linear Gaussian systems, where it is assumed that errors and white process noise are distributed normally.

What exactly does a filter do, and what happens to it when outliers are present?

Mathematically, a filter is a multivariate representation that defines the state of the system. Systems are represented as models that are the underlying schematics of the filter. In the present consideration, systems or models are defined by the linear relation of the current process value to previous process values using autoregressive parameters as the coefficients. These coefficients are not known, but must be estimated from the data. This type of estimator is directly connected to the data, so any problems with these estimators affect the performance of the filter that may rely on them. Hence filters are functions of estimators that can be removed from the data by different degrees of removal. Herein lies the problem. Outliers may throw off the performance of the initial or primary estimates in a filter. In this thesis, we examine the Naive filter, which is thought to be robust to poor parameter estimates.

1.1 The Idea of Robustness

Robust estimators or statistics are those that estimate a true quantity with a suitable level of resistance to anomalous events. As a demonstration of robustness issues, the common mean is now shown to lack robustness. Consider the following randomly generated data from a standard normal distribution given in the vector below.

$$\boldsymbol{x} = \begin{pmatrix} 0.5061 \\ -1.0621 \\ -0.6185 \\ 0.1229 \\ 0.6704 \\ -1.4243 \\ 0.7412 \\ 0.5173 \\ 0.7783 \\ -0.5474 \end{pmatrix} \tag{1.1}$$

The mean of this data vector is $\bar{x} = -0.03161$, which seems a reasonable approximation to the true mean of zero. If an outlier is injected into the data from a different distribution, what would happen to this estimate of the true mean? We add the value 12 and the mean becomes $\bar{x} = 1.062173$. This description can be made more mathematical using the sensitivity curve defined by Maronna et al. (2006).

$$\hat{\mu}\left(x_1, x_2, \ldots, x_n, x_o\right) - \hat{\mu}\left(x_1, x_2, \ldots, x_n\right) \tag{1.2}$$

Here the effect of the outlier x_o is measured using this sensitivity curve, where we vary the value x_o across the number line. If an estimator is robust, the difference between the values given in equation (1.2) will be small for a wide range of x_o values. Non-robust estimators will have a small or non-existent range where this difference is small. Hence, robustness is demonstrated in estimators when they exhibit small differences from data without outliers to data containing outliers.

1.2 Robust Filters

This thesis examines the construction of a robust filter. Robust filters are needed to process the many types of data that exhibit a wide range of outlier problems. Such problems in time series data can grossly confound filters. For this reason, robustness is a must in the control world. Another circumstance that contributes to this need is in the way the filter is used in control. An *in-loop* or real time use of filters makes robustness a necessity. These processes are not watched by process controllers and adjusted based on graphs and tables of numbers—they are designed to control a process in a more fully automated fashion.

One example that illustrates this type of control is in the control of an inverted pendulum, where a weight is positioned at the end of a rod with a swivel connecting the rod to a cart. To keep the pendulum from falling, a good control algorithm is needed. A robust filter is a key element in a control algorithm; it estimates what the state of the system is and passes this information to the algorithm for control action determination.

1.3 Kalman Type Filters

The Kalman filter is one approach that can be scrutinized for robustness and used as a benchmark for the estimation of state. Based on the recursive least squares estimator, it seems to inherit statistical properties that rank it average for dealing with robustness issues. A filter is only as good as its underlying estimators, and hence the Kalman filter is subject to these underlying estimates of autoregressive and impulse response parameters.

1.4 Proposed Use of Markovian Representation

This thesis examines the derivation and empirical validation of the Naive filter for control of dynamic systems. An in-depth derivation of the Naive filter and its use in state estimation is given. Comparisons of different robust filters along with the much used Kalman filter are used. The Naive filter itself is extremely susceptible to measurement error and relies heavily on a method to remove this type of error.

The model itself is an interesting mixture of time series parts. The autoregressive part of the model is not necessarily stationary, but can use meaningful input adjustments to bring the process into control, or to a stationary state. In typical use, state estimates are used in parallel with a controller such as the Linear Quadratic Regulator (LQR), which gives a controlling input to force a process to point at a target. The impulse response piece simply weights the inputs, whether they are just exogenous inputs or controlling inputs. This is done in control to help dampen the control inputs and keep the controller stable.

A state estimate is required when a process contains much measurement error. The Kalman type filters need at least four different steps to estimate the state. Each of these steps requires estimates of the process's time series parameters. If these estimates are poor, then the poor estimates are used at least four times by the filter before the state is estimated. Akaike (1974) deals with a hypothetical state vector he calls the Markovian representation that provides an idea for a state estimate that uses the poor estimates only once. This has supported the idea to create a more time series based estimate that uses the autoregressive structure in the estimate as the primary source of the estimate. It will be shown to rival the most robust filters considered. This estimate is especially useful in control, not just in estimation alone.

The Krein space Kalman filter is a very good filter if its robust functionality is optimized. This thesis proposes two enhancements that greatly maximize the filter's ability to estimate state.

2. LITERATURE REVIEW

For simplicity, consider the model of the single input single output (SISO) system given in (2.1) below, which models an autoregressive impulse response time series ARIR(p,q). This specific time series model will be discussed in more detail in the next section. An SISO system of this sort explains one value from a process that could be defined by all past values in an autoregressive manner, as in Equation (2.2), along with a linear combination of some exogenous inputs. The process error, ϵ_n, is a quantity that also defines the next value coming from the process. One could consider the multiple input multiple output (MIMO) system; however, its complexity is unnecessary to discuss in this presentation.

$$\begin{aligned} z_n + \phi_1 z_{n-1} + \phi_2 z_{n-2} + \cdots + \phi_p z_{n-p} = \\ \theta_1 u_{n-1} + \theta_2 u_{n-2} + \cdots + \theta_q u_{n-q} + \epsilon_n \end{aligned} \tag{2.1}$$

$$\begin{aligned} z_n = -\phi_1 z_{n-1} - \phi_2 z_{n-2} - \cdots - \phi_p z_{n-p} + \\ \theta_1 u_{n-1} + \theta_2 u_{n-2} + \cdots + \theta_q u_{n-q} + \epsilon_n \end{aligned} \tag{2.2}$$

This system is meant to model processes that are possibly out of control due to parameter values in ϕ_i, $i = 1, \ldots, p$, that are not in a region where stationarity holds, where z_{n-i} and u_{n-h} are scalars, for $h = 1, 2, \ldots, q$. These scalars symbolize the data and the exogenous inputs. In control applications, the impulse response (IR) part is a special control action added to stabilize out of control tendencies of the autoregressive (AR) part and to compensate for outside influences on the system. The ulimate objective of controlling a process is to make it look like a normal autoregressive process with no moving average. A target must be defined about which the random process error will vary. We would like it to look like the following graph.

Figure 2.2 is a demonstration of an ARIR(2,2) with the same error values as the previous AR(2) time series, only with "wild" exogenous inputs that drive the time series process off target.

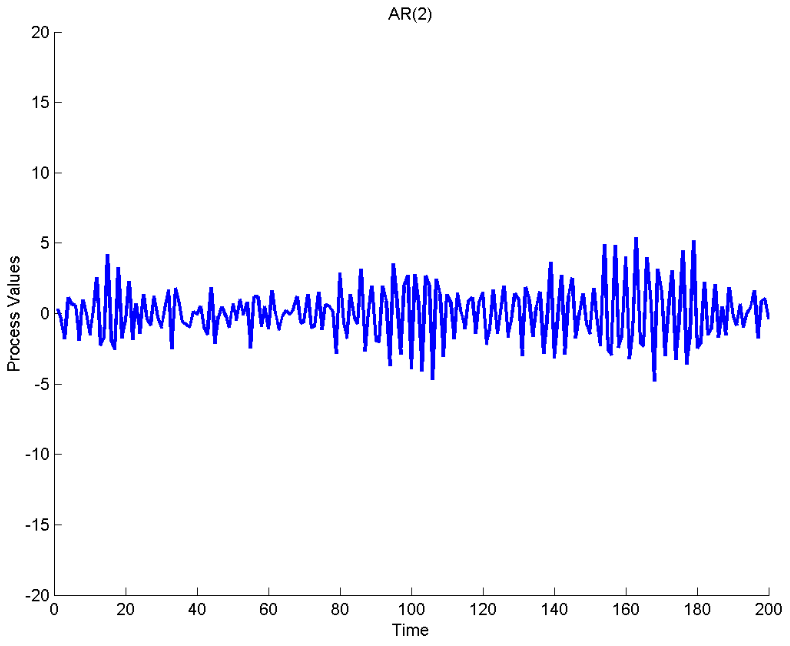


Figure 2.1: Example of an on-target time series.

To set the stage for the review of existing literature, similar notation as that found in Franklin et al. (1998) will be used to define the variables. The following are definitions of the system variables in Equations 2.3 and 2.4, which are defined in the state space representation.

The vector $\boldsymbol{x_n}$ is the state of the system, which contains our primary source for examining what the real process values are, and $\boldsymbol{y_n}$ is the measured values that contain measurement error. Since there is measurement error in the system, the exact observance of the process value, z_n, is impossible; hence Equations 2.1 and 2.2 are only theoretical relations that are never truly known. The following equations define the state space model and the variables in the model:

$$\boldsymbol{x_n} \quad = \quad \boldsymbol{A x_{n-1} + B u_{n-1} + G \epsilon_n}, \tag{2.3}$$

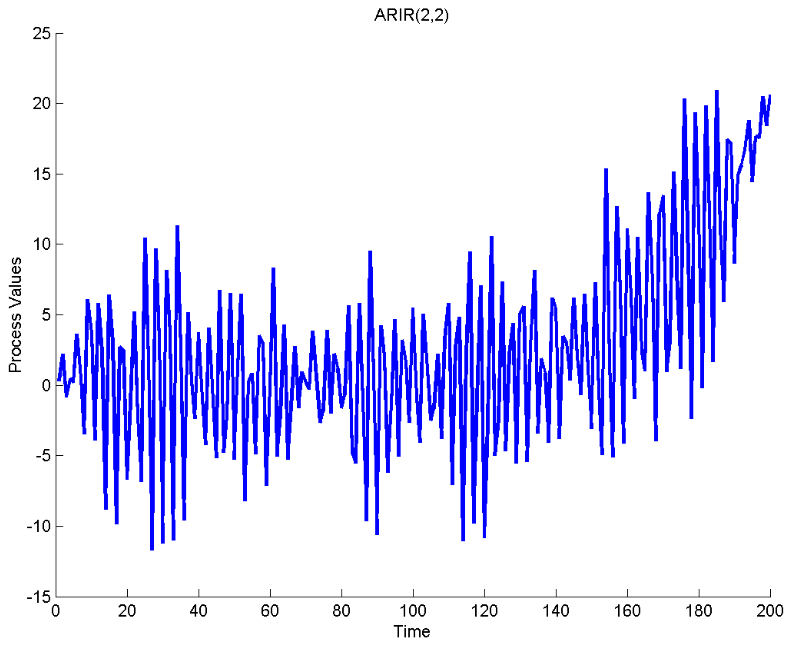


Figure 2.2: Example of an off-target time series.

$$\boldsymbol{y_n} = \boldsymbol{C}\boldsymbol{x_n} + \boldsymbol{\eta_n}. \tag{2.4}$$

Define $d = max(p, q)$ as the maximum of the two orders p and q; n is the index used for time,

- $\boldsymbol{x_n}$ is $d \times 1$,
- $\boldsymbol{y_n}$ is 1×1 ($d \times 1$ for the Naive filter),
- $\boldsymbol{\epsilon_n} \sim N(\boldsymbol{0}, \boldsymbol{V_\epsilon})$ is $d \times 1$ and $\boldsymbol{V_\epsilon} = \sigma_\epsilon^2 \boldsymbol{I}$,
- $\boldsymbol{\eta_n} \sim N(\boldsymbol{0}, \boldsymbol{V_\eta})$ is 1×1 ($d \times 1$ for the Naive filter) and $\boldsymbol{V_\eta} = \sigma_\eta^2 \boldsymbol{I}$,
- $\boldsymbol{u_n}$ is a scalar,
- $\boldsymbol{A}$ is $d \times d$,
- $\boldsymbol{B}$ is $d \times 1$,
- $\boldsymbol{C}$ is $1 \times d$,
- $\boldsymbol{G}$ is $d \times d$.

The matrix $\boldsymbol{A}$ defines the dynamics of the system. The structure of this matrix is specific as defined by Maronna et al. (2006) and will work for the Kalman, Naive, and other Robust filters. The special structure is

$$\boldsymbol{A} = \begin{pmatrix} \phi_1 & 1 & 0 & \cdots & 0 & 0 \\ \phi_2 & 0 & 1 & & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ \phi_{d-1} & 0 & 0 & & 0 & 1 \\ \phi_d & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}. \tag{2.5}$$

When we have a multiple input multiple output (MIMO) system, the $\boldsymbol{A}$ matrix is block diagonal with the structure in Equation 2.5 above.

Matrix $\boldsymbol{B}$ contains weights, θ, of exogenous actions or inputs, whether control or from independent random processes. The representation,

$$\boldsymbol{B} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{pmatrix}, \tag{2.6}$$

describes in part how much these exogenous inputs alter the linear system. Notice that the dimensions of this matrix, as well as the dimension of $\boldsymbol{A}$, are not in terms of q but of d, which is the maximum of p and q. The reasoning here is based on whether there is a higher order for the autoregressive part of the system or for the impulse response part; hence, if p is the maximum between p and q then matrix $\boldsymbol{B}$ will be padded with zeros where we don't have parameters, and vice versa for $\boldsymbol{A}$ when q is greater than p. If this were an ARIR(4,2) model where there are only two IR parameters estimated, the vector $\boldsymbol{B}$ in this case will be padded with two zeros to get the appropriate dimension.

Matrix $\boldsymbol{A}$ coupled with the state vector defines how the dynamics change through time. The state vector $\boldsymbol{x_n}$ is relative to the Markovian representation defined by Akaike (1974). This Markovian representation is related to the process values z_n in the following manner:

$$\boldsymbol{\hat{x}_n} = \begin{pmatrix} \boldsymbol{\hat{z}_{n|n}} \\ \boldsymbol{\hat{z}_{n+1|n}} \\ \vdots \\ \boldsymbol{\hat{z}_{n+d|n}} \end{pmatrix}. \tag{2.7}$$

Here the predictions for the d-step state vector are estimated as

$$\boldsymbol{\hat{z}_{n+i|n}} = \sum_{j=i}^{d-1} \boldsymbol{-\Phi_j z_{n-j+(i-1)} + \Theta_j u_{n-j+(i-1)}} \ \textit{and} \tag{2.8}$$

$$\boldsymbol{\hat{z}_{n|n}} = \boldsymbol{z_n}, \tag{2.9}$$

where the index i is now defined for $i = 1, \ldots, d$. This structure has a kind of diminishing representation as far as its use of the data and inputs. Akaike (1974) reveals the somewhat obscure idea of the state of a system. The state gathers as much information from the available data as possible. The Markovian representation will be useful in examining use of all of this information. One thing to consider is that the Kalman filter uses only a small portion of the information furnished by the state, as seen in the $\boldsymbol{C}$ matrix. For the SISO Kalman filter the $\boldsymbol{C}$ matrix is

$$\boldsymbol{C} = \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}. \tag{2.10}$$

Only the first element is used based on Equation (2.4). It is possible to use this additional information that is inherent to the state vector. The $\boldsymbol{C}$ matrix is set equal to the identity matrix.

This Markovian representation does not account for any measurement error. At this point, it must be recognized that the estimate of state given in Equations 2.8 and 2.9 is not obtainable in engineering applications. The measured values are actually of the form

$$\boldsymbol{z_n^*} = \boldsymbol{z_n} + \boldsymbol{\eta_n}. \tag{2.11}$$

This is necessary to realize because the use of the exact values z_n is not possible, so z_n^* must be used instead.

To return to the description of the system variables in Equations 2.3 and 2.4, we have matrix $\boldsymbol{G}$ which represents the system correlation and effectively structures how the process error comes in to the state space model. The process error $\boldsymbol{\epsilon_n}$ is a Gaussian process of white noise.

The output vector $\boldsymbol{y_n}$ is a scalar for the Kalman filter but is a $d \times 1$ vector for the Naive filter; notice Equation 2.4. The Kalman state vector uses y_n as the new data for an update. The Naive filter will use this same piece of information and will also glean any possible information out of past values of the additional Markovian terms found in (2.7).

Measurement error enters the system through $\boldsymbol{\eta_n}$, which is a Gaussian process that represents the measurement discrepancy for each process measured. The matrix $\boldsymbol{C}$ relates what the state is and how we really measure it. The state can be directly measured and would be modeled using $\boldsymbol{C} = \boldsymbol{I}$, the identity matrix after which the measurement error is incorporated into the system.

2.1 Recursive LS Type Filters

2.1.1 Kalman Filter

In this section, Kalman filtering will first be presented as an estimate to get the fundamentals of the filter down. Then the prediction form of the Kalman filter will be presented.

The Kalman filter is founded on least squares estimation (see *Digital Control of Dynamic Systems* Franklin et al. (1998)). The recursive least squares estimator presented by Franklin is given, along with the derivation from the ordinary form for linear regression. This estimator is the precursor to the Kalman filter. The Kalman filter is a recursive type estimator that has a possible problem with robustness because it has four updates every time there is a new piece of data obtained (Franklin et al. 1998). There are two time updates and two measurement updates. If there is a problem estimating the parameters in matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, then this error in estimation is compounded all four times we use them.

These updates are presented below in a sequential manner. These estimators are now denoted using different notation to designate the estimators' dependence on past or present data; that is, $n|n-1$ or $n|n$, respectively.

$$\boldsymbol{P_{n|n-1}} = \boldsymbol{A P_{n-1|n-1} A^T} + \boldsymbol{G V_\epsilon G^T} \tag{2.12}$$

$$\boldsymbol{P_{n|n}} = \left(\boldsymbol{P_{n|n-1}^{-1}} + \boldsymbol{C^T V_\eta^{-1} C}\right)^{-1} \tag{2.13}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}} + \boldsymbol{B}\boldsymbol{u}_{\boldsymbol{n-1}} \tag{2.14}$$

$$\boldsymbol{L}_{\boldsymbol{n}} = \boldsymbol{P}_{\boldsymbol{n|n}}\boldsymbol{C}^{\boldsymbol{T}}\boldsymbol{V}_{\boldsymbol{\eta}}^{-1} \tag{2.15}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{L}_{\boldsymbol{n}}\left(\boldsymbol{y}_{\boldsymbol{n}} - \boldsymbol{C}\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}\right) \tag{2.16}$$

The first of these updates, Equation (2.14), is the time update for the state estimator in the Kalman filter. It utilizes the dynamics of the system and the control action weights found in matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Again, the subscript denotes an estimate of the state of the system for time step n given the data $\{y_1, \ldots, y_n\}$. In dynamic control it is important to predict the future state from this estimate so that a control action can be found to apply next time step.

The estimate of the current state based on past data has the covariance matrix $\boldsymbol{P}_{\boldsymbol{n|n-1}}$, which is needed in the measurement update for the state estimator. This is a time update only for the covariance. In effect, it is a prediction of the covariance for the next time step. Measurement updating starts with the covariance matrix $\boldsymbol{P}_{\boldsymbol{n|n}}$, which is the covariance matrix for the state measurement update in Equation 4.2. In this update the difference between the predicted $\hat{\boldsymbol{y}}_{\boldsymbol{n}} = \boldsymbol{C}\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$ and the measured values in the true vector are used to adjust for the error in the time update prediction of the state.

In Equation 4.2, there is an estimation gain matrix $\boldsymbol{L}_{\boldsymbol{n}}$ that varies with time. Since prediction is the desired goal, the gain matrix will now be thought of as a predictive gain matrix in the following fashion:

$$\boldsymbol{L}_{\boldsymbol{P,n}} = \boldsymbol{A}\boldsymbol{L}_{\boldsymbol{n}}. \tag{2.17}$$

The previous equations were presented in a manner that gave the order of the use of the equations for Kalman estimation. In the equations below, there is a presentation of the order the equations need to be done for a prediction of state as in Franklin et al. (1998).

$$\boldsymbol{L}_{\boldsymbol{P,n}} = \boldsymbol{A}\boldsymbol{P}_{\boldsymbol{n|n-1}}\boldsymbol{C}^{\boldsymbol{T}}\boldsymbol{V}_{\boldsymbol{\eta}}^{-1} \tag{2.18}$$

$$\hat{\boldsymbol{x}}_{n+1|n} = \boldsymbol{A}\hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{B}\boldsymbol{u}_n + \boldsymbol{L}_{P,n}\left(\boldsymbol{y}_n - \boldsymbol{C}\hat{\boldsymbol{x}}_{n|n-1}\right) \tag{2.19}$$

$$\boldsymbol{P}_{n|n} = \left(\boldsymbol{P}_{n|n-1}^{-1} + \boldsymbol{C}^T\boldsymbol{V}_\eta^{-1}\boldsymbol{C}\right)^{-1} \tag{2.20}$$

$$\boldsymbol{P}_{n+1|n} = \boldsymbol{A}\boldsymbol{P}_{n|n}\boldsymbol{A}^T + \boldsymbol{G}\boldsymbol{V}_\epsilon\boldsymbol{G}^T \tag{2.21}$$

Notice that the time update for the state has been incorporated into the measurement equation. The first computation invoked is the prediction of the state, and then the covariance matrices are computed last in preparation for the next step in time.

2.1.2 Krein Space Kalman Filter

The Krein space is an indefinite inner product space. In Hassibi et al. (1996a) the author explains that linear estimation in Krein space is "richer than that of the conventional Hilbert space." A Hilbert space is designed to find the minimum of a quadratic cost function using projection methods. However, the Krein space, using a projection, is designed to make the quadratic forms stationary as stated in Lee et al. (2004). Lee makes it apparent that the the Kalman filter will offer a "state estimation strategy" instead of an actual minimum point. As will be detailed in this section, certain conditions must exist for this strategy for robust estimation in the Krein space to work. For more information on these topics, consult the cited articles.

The strategy here is not a plug and chug method. It requires the knowledge or at least the estimation of the uncertainty. There are a number of solutions that are admissible in the Krein space, and one must concentrate on finding one of them to achieve the linear estimation available through use of this space with projections. The end result of the quadratic cost function is not a minimum point, but a closed set of possible solutions. In Equation 2.31 we see the description of the solutions. As the build up of this filter unfolds, notice that the approach in this thesis will be involved with finding solutions on the boundary of these sets.

Lee et al. (2004) investigates the use of the Krein space Kalman filter for state estimation. The purpose is to find a Kalman type filter that is robust to poor parameter estimates in the transition matrix $\boldsymbol{A}$. This accounting of the inability of matrix $\boldsymbol{A}$ to estimate parameters is thought of as uncertainty in the system. We will represent this uncertainty for each of the parameters as δ_{ϕ_i} for uncertainty in ϕ_i, and δ_{θ_j} for the uncertainty in θ_j, where $i = 1, \ldots, p$ and $j = 1, \ldots, q$, with p and q being the order of the system. This requires the ability to estimate uncertainty. In the paper by the above authors they consider the uncertainties as known quantities for the sake of demonstrating the effectiveness of the filter to handle poor parameter estimates. In this document a modification of this approach will be taken—that of representing our knowledge of the uncertainties as coming from a sampling distribution. The values of δ_{ϕ_i} and δ_{θ_j} will be jittered with random noise to simulate the inability to perfectly estimate uncertainty.

A study of simple projections along with Hilbert spaces followed by Krein space estimation with respect to quadratic terms will reveal the philosophy behind this method. There is a condition that must be met to prove that the estimation is indeed part of the Krein space estimation. As explained below, the uncertainties begin with the following state and measurement equations. As a side note, the exogenous input term from the true state, that is $\boldsymbol{Bu_n}$, is actually in the error term $\boldsymbol{\zeta_{n+1}}$. This will be derived below.

$$\boldsymbol{x_{n+1}} = \boldsymbol{A_2 x_n} + \boldsymbol{G_1 \Delta_{\delta_1} K x_n} + \boldsymbol{G \zeta_{n+1}} \tag{2.22}$$

$$\boldsymbol{y_n} = \boldsymbol{C x_n} + \boldsymbol{\eta_n} \tag{2.23}$$

Lee et al. (2004) set the uncertainty equal to a term $\boldsymbol{s_n} = \boldsymbol{K x_n}$; later this term will be very important in choosing the values that go inside the matrices in (2.22) above.

The above state equation can be derived from the state equation found in (2.3). This equation contains all the same information as that found in (2.22). Consider the transition matrix $\boldsymbol{A_\Delta}$. This matrix has inherent uncertainty and as such,

$\boldsymbol{A_\Delta} = \boldsymbol{A} + \boldsymbol{\Delta}$, where the following relations hold,

$$\begin{aligned}
\boldsymbol{\Delta} &= \begin{pmatrix} -\delta_1 & 0 \\ -\delta_2 & 0 \end{pmatrix} && (2.24)\\
&= \begin{pmatrix} -\delta_1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -\delta_2 & 0 \end{pmatrix} && (2.25)\\
&= \boldsymbol{\Delta_1} + \boldsymbol{\Delta_2}. && (2.26)
\end{aligned}$$

The Krein space Kalman filter does not account for the exogenous inputs in the model, so for the uncertainty state space model the exogenous part sinks into the process error term.

$$\begin{aligned}
\boldsymbol{x_{n+1}} &= \boldsymbol{A_\Delta x_n} + \boldsymbol{B u_n} + \boldsymbol{G \epsilon_{n+1}}\\
&= \left(\boldsymbol{A_{\Delta_2}} + \boldsymbol{\Delta_1}\right)\boldsymbol{x_n} + \boldsymbol{G}\left(\boldsymbol{G^{-1} B u_n} + \boldsymbol{\epsilon_{n+1}}\right)\\
&= \boldsymbol{A_{\Delta_2} x_n} + \boldsymbol{G_1 \Delta_{\delta_1} K x_n} + \boldsymbol{G \zeta_{n+1}}
\end{aligned}$$

Here the value is denoted for the new process error.

$$\boldsymbol{\zeta_{n+1}} = \boldsymbol{G^{-1} B u_n} + \boldsymbol{\epsilon_{n+1}} \qquad (2.27)$$

In the above derivation it is sufficient but not necessary for the following equalities to exist for the filter defined to be in the Krein space, heuristically speaking.

$$\begin{aligned}
\boldsymbol{\Delta_1} &= \boldsymbol{G_1 \Delta_{\delta_1} K} && (2.28)\\
\boldsymbol{G_1} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} && (2.29)\\
\boldsymbol{K} &= \begin{pmatrix} \delta_1 & 0 \end{pmatrix} && (2.30)
\end{aligned}$$

There is a condition that must be met in order for the filter to be part of the Krein space. First we set $\boldsymbol{\xi_n} = \boldsymbol{\Delta_{\delta_1} K x_n}$.

$$\sum_{n=0}^{N} \|\boldsymbol{\xi_n}\|^2 \leq \sum_{n=0}^{N} \|\boldsymbol{s_n}\|^2 \qquad (2.31)$$

The vectors in (2.31) are parts of the $\Re^1$ space. Taking the vector norm of a vector in this space is simply the absolute value, and the matrix $\boldsymbol{\Delta_{\delta_1}}$ proves to be the only term that distinguishes $\boldsymbol{\xi_n}$ from $\boldsymbol{s_n}$.

$$\boldsymbol{\xi_n} = \boldsymbol{\Delta_{\delta_1} K x_n} = \boldsymbol{K x_n} = \boldsymbol{s_n} \quad if\ |\boldsymbol{\Delta_{\delta_1}}| = 1 \tag{2.32}$$

By setting $\boldsymbol{\Delta_{\delta_1} = -1}$, the condition is met. This value for $\boldsymbol{\Delta_{\delta_1}}$ establishes (2.28) and gives the boundary solution as well.

The estimator is as follows (Lee et al. 2004), given the notation expressed above in (2.28 - 2.30) and that found in the beginning of this chapter.

$$\boldsymbol{\hat{x}_{n+1|n}} = \boldsymbol{A_\Delta \hat{x}_{n|n-1}} + \boldsymbol{L_n} \begin{bmatrix} \boldsymbol{y_n - C\hat{x}_{n|n-1}} \\ \boldsymbol{-K\hat{x}_{n|n-1}} \end{bmatrix} \tag{2.33}$$

$$\boldsymbol{L_n} = \boldsymbol{A_\Delta P_{n|n-1} \tilde{C}^T V_{\eta_\delta}^{-1}} \tag{2.34}$$

$$\boldsymbol{V_{\eta_\delta}} = \boldsymbol{\tilde{C} P_{n|n-1} \tilde{C}^T + \tilde{V}_\eta} \tag{2.35}$$

$$\boldsymbol{\tilde{V}_\eta} = \begin{pmatrix} \boldsymbol{V_\eta} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{-I} \end{pmatrix} \tag{2.36}$$

$$\boldsymbol{V_{\epsilon_K}} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{V_\epsilon} \end{pmatrix} \tag{2.37}$$

$$\boldsymbol{\tilde{C}} = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{K} \end{pmatrix} \tag{2.38}$$

$$\boldsymbol{P_{n+1|n}} = \boldsymbol{A_\Delta P_{n|n} A_\Delta^T + G_K V_{\epsilon_K} G_K^T} \tag{2.39}$$

$$\boldsymbol{G_K} = \begin{pmatrix} \boldsymbol{G_1} & \boldsymbol{G} \end{pmatrix} \tag{2.40}$$

$$\boldsymbol{P_{n|n}} = \boldsymbol{P_{n|n-1} - P_{n|n-1} \tilde{C}^T V_{\eta_\delta}^{-1} \tilde{C} P_{n|n-1}}. \tag{2.41}$$

It can be noticed in the equations above that this has the general form of the Kalman filter state estimate. However, it has many changes to accommodate the uncertainty adjustments.

2.1.3 Minor Additions and Adjustments to the Krein Kalman

The information in this section is not necessarily a review of literature, but rather an extension to it. The applications created here are extensions to what has been done in the paper written by Lee et al. (2004). It seems many authors have put much consideration on strictly autoregressive processes without as much consideration on exogenous inputs. The additional information presented here is included in this chapter in order to keep it with the original description of the Krein space Kalman filter.

The Krein space Kalman filter does not have any terms that account for the exogenous inputs as does the classic Kalman filter. This would be a very simple adjustment that may prove to be useful for increasing robustness. Another shortcoming of this filter is that it does not adjust for possible uncertainty in the second parameter in an ARIR(2,2) system. To add an adjustment for the second uncertainty parameter, it is necessary to change the selected matrices as follows:

$$\boldsymbol{K} = -\boldsymbol{\Delta}, \tag{2.42}$$

$$\boldsymbol{G_1} = \boldsymbol{I}, \ \textit{and} \tag{2.43}$$

$$\boldsymbol{\Delta_\delta} = -\boldsymbol{I}. \tag{2.44}$$

The matrix $\boldsymbol{\Delta_\delta}$ is the counterpart of $\boldsymbol{\Delta_{\delta_1}}$ in the previous section. Then, to adjust to account for the exogenous inputs, $\boldsymbol{Bu_n}$ is simply added to Equation (2.45).

$$\boldsymbol{\hat{x}_{n+1|n}} = \boldsymbol{A_\Delta \hat{x}_{n|n-1}} + \boldsymbol{Bu_n} + \boldsymbol{L_n} \begin{bmatrix} \boldsymbol{y_n} - \boldsymbol{C\hat{x}_{n|n-1}} \\ -\boldsymbol{K\hat{x}_{n|n-1}} \end{bmatrix} \tag{2.45}$$

If one does the math to check the condition in (2.31) in the previous section, the condition will again result in an equality, thus satisfying the condition. Note the reason below, which is in parallel to that satisfying the condition in the previous section.

$$\|\boldsymbol{\Delta_\delta}\| = |-1|\|\boldsymbol{I}\| \tag{2.46}$$

The matrix norm of the identity matrix equals one; hence the inequality of the condition (2.31) is once again met just as in the original Krein space Kalman filter setup in the last section, only for an increased space. In the end, there will be three versions of the Krein space Kalman filter to compare against the derived filter of this thesis. These three are the original Krein space Kalman filter, the Fully Modeled Krein space Kalman filter, and the Fully Modeled Autoregressively Adjusted Krein space Kalman filter. The last two are abbreviated as the F.M. Krein space Kalman filter and the F.M.A.A. Krein space Kalman filter respectively.

The first filter, the F.M. filter, is termed the Fully Modeled Krein space Kalman filter because it models all parts of control applications—the autoregressive and impulse response parts. The F.M.A.A. Krein space Kalman filter performs the full modeling as well as adjusting for the second uncertainty parameter. It has the additional name *Autoregressively Adjusted* because all the uncertainty with respect to all the autoregressive parameters is taken into account.

2.2 Outliers

Certain types of outliers, such as those discussed in *Robust Statistics* (Maronna et al. 2006), are of importance in examining the effective robustness of the Kalman filter. These are isolated outliers, level shift outliers, and patch outliers. They can be presented in many different ways. For instance, we can see them brought into the system through the measurement error; these are called additive outliers (AO). AOs are usually isolated outliers, especially in the case we are considering where the measurement error is iid. These AOs are always possible in the process; that is, at every time step we have an AO. The variance of the measurement error is expected to be much larger than the process error variance, $\sigma_\eta^2 >> \sigma_\epsilon^2$. However, if we get one that is abnormally large it will cause problems.

Patch outliers can be exhibited in the additive form as well. Patch outliers occur happen when there is a small run of outliers in a row. One form of these are doublets that cancel each other out.

Replacement outliers (RO) can lead to new avenues for filters, such as regime switching filters, where we have data from many different distributions in the same system. A Markov model can be employed to switch between various models with unique parameters. Hence, we build robustness into the filter by modeling all these different ROs with an assortment of models if we have reason to believe there are such outliers present. This requires the assumption that these unique regimes are reasonably regular in their expression over time. The robustness that a filter of this form would inherit appears to be phenomenal.

Outliers can also come from the process error. In this case they are referred to as innovation outliers, where we see a propagation of the outlier down through the process. That is, the effect of the innovation outlier will die out after a certain number of time steps. This is a direct result of the autoregressive nature of this time series. One method of modeling a process error is with a mixture distribution for the process error, such as $(1-\omega)\,N\left(0,\sigma_{\epsilon 0}^2\right)+\omega N\left(0,\sigma_{\epsilon 1}^2\right)$, where ω is small.

3. PROBLEM AND SOLUTION DESCRIPTION

3.1 The Problem of Poor Parameter Estimates

In control of dynamic systems, it is of the utmost importance to correctly estimate the state of the system. If the state is poorly estimated, the control may be poor as well. Commonly used filters that estimate the state of the process have been shown mathematically to be lacking in robustness and to provide poor information in determination of the control action at times (Doyle 1978).

3.2 Use of the Markovian Representation

To solve the problem of poor state estimation, a naive robust filter will be derived in full and evaluated to show its robustness properties in comparison to other robust filters and the classic Kalman filter. This Naive filter is expected to have some sensitivity to measurement error. For this reason, this Naive filter will be preceded by a smoother as a pre-filter. To evaluate the filter, one needs to simulate a system and add measurement error to it, then compare what the known truth is with the estimated state using the filter. If the estimated state matches perfectly with the simulated system, excluding the measurement error, then the filter has perfectly modeled the system. In this situation it is hoped that a filter can portray the correct state information to a controller in order to keep it on target. In many applications the target is zero. In Figure (2.2) an in-control system is demonstrated.

To set the stage for this treatment of the Naive filter, the idea of robustness is expanded upon in a more defined manner, separating the robustness problems into three distinct arenas. This discussion is based on personal adaptations of the current problem inspired by the literature review. The following topics have resulted from study of the literature reviewed and represent simple developments to support the ideas in this thesis.

3.3 Different Degrees of Robustness

In this section, the phrase, *removal from data*, is a term that describes the dependence of an estimator on the data. Consider the filter as an estimator. It is a function of not only the data, but of other estimators. There are many things to consider to reach the ultimate goal of controlling a process. The first objective is to find the order of the autoregressive time series, or in other words, to find the number of meaningful parameters necessary to parametrize the model.

Next, the parameters themselves need to be estimated from the data. The value of these estimates depends on how many parameters were fit, which is a direct result of finding the order of the time series. The estimate of the parameters is a function of the data and the order.

For simplicity, consider the difference in the estimated mean and variance of a random variable.

$$\bar{x}(\mathbf{x}) = \frac{\sum_{i=1}^{n} x_i}{n} \quad \textit{and} \tag{3.1}$$

$$\hat{\sigma}^2(\mathbf{x}, \bar{x}) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}. \tag{3.2}$$

As common knowledge dictates, the variance is a function of the data and the mean, where the mean is only a function of the data. The variance has a higher level of removal from the data.

The presentations in the literature review show that there are many levels of estimates that are more or less removed from the data. Removal from the data is represented in this sequence: the base parameters ϕs and θs are estimated from the data directly, and filters are not only functions of the data but of the parameters ϕ and θ. Hence, the filter is more removed from the data because more of the information used by the filter is derived from these parameters. If the parameters are poorly estimated the effect will be expressed in the performance of the filter that is removed from the data using these parameters.

Having discussed removal from data, it is noted here that the focus is on a higher level of removal from the data. As presented earlier, there are those estimates that, like the mean, are directly related to the data. Others are functions of more than just the data, but of many parameters. The robustness, or lack thereof, will play in to the ability of the overall control of the process. If the variance is a poor estimate of dispersion of the data, then it is more than likely poor because of the mean, which is very inadequate when outliers are present. The filter that is sought here is robust not only to the data, but to poor estimators.

Heuristically speaking, the effect of outliers will cause an underestimation of the autoregressive parameters because the outliers will be far from the rest of the data. These points will cause the estimate of the autocorrelation, which is a component of the autocovariance, to be smaller. Underspecification of the model could occur, and hence bias increases and variance decreases. The ARIR parameter estimates themselves will also suffer in like manner and the effect will flow down through the different stages of the controller.

3.3.1 High Level Robustness

In the controller there are terminable estimates that are the final steps before the control action is found. Estimators that are the furthest removed from the data will suffer high level robustness problems. They are based on functions of low level estimators as well as the original data. These estimators are much more complex in structure and computation, so more error is inherent.

A filter can also have a certain level of immunity to this through pure mathematical structure. The Naive filter is thought to have a more natural robustness property, although it is not apparent at this point which robustness benefits it has. Since the Kalman filter is derived from the recursive least squares estimator, it should inherit all the properties that are found in the normal least squares estimator, which is not particularly robust.

3.3.2 The Naive Filter

The Naive filter is similar to the robust state vector discussed in the section on the Markovian Representation. The Naive filter is suspected to have a pure natural structure that handles both outlier data coming in to the filter and poor estimates of the system parameters without the rudimentary adjustments.

This is a filter that is similar in mathematics to what was proposed by Akaike (1974) in the Markovian representation. It portrays the idea of gathering as much information as possible to estimate the state of the process. Here the estimator attempts to predict into the future as far as the data will allow, given the order of the system. The representation is

$$\hat{x}_{n+1|n} = \begin{pmatrix} \hat{z}_{n+1|n} \\ \hat{z}_{n+2|n} \\ \hat{z}_{n+3|n} \end{pmatrix}. \tag{3.3}$$

The form of this filter is given here for a simple ARIR(2,2) case. It is very much like the structure of the autoregressive moving average ARMA(2,2) but has distinct differences in the underlying mathematics. The Naive state vector is defined for the simple autoregressive impulse response of order (2,2). This vector is used in the state equation (2.3) as the estimate of the previous state and is then used in the state equation to predict what the state will be next time step.

$$\boldsymbol{\hat{x}_{n+1|n}} = \begin{pmatrix} -\phi_2 z^*_n - \phi_3 z^*_{n-1} + \theta_2 u_n + \theta_3 u_{n-1} \\ -\phi_3 z^*_n + \theta_3 u_n \end{pmatrix} \tag{3.4}$$

This equation is defined in this manner for use in real-world applications, to be able to take the raw data and the control inputs and calculate the necessary state estimate as defined in the Naive filter (3.4). The matrices $\boldsymbol{T_z}$ and $\boldsymbol{T_u}$ are designed with the structure found in Equations 3.6 and 3.7 to create the structure found in the vector equation, (3.4), and can be thought of as the generating function of the state vector for use in the state equation, (2.3). The generating function is

$$\boldsymbol{\hat{x}_{n|n}} = \boldsymbol{T_z z_n^*} + \boldsymbol{T_u u_n}, \tag{3.5}$$

where $\boldsymbol{z_n^*} = \begin{pmatrix} z_n^* & z_{n-1}^* & \dots & z_{n-d}^* \end{pmatrix}^T$, with $z_n^* = z_n + \eta_n$, is the data vector which has memory of d data observations and $\boldsymbol{u_n} = \begin{pmatrix} 0 & u_{n-1} & \dots & u_{n-d} \end{pmatrix}^T$ is set up the same with the first element as zero and the current control action u_n added into the estimate in the state equation.

Here is a description of the structure of both $\boldsymbol{T_z}$ and $\boldsymbol{T_u}$ (remember that $d = max\,(p, q)$).

$$\boldsymbol{T_z} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & -\phi_1 & \cdots & -\phi_{d-1} & -\phi_d \\ 0 & -\phi_2 & \cdots & -\phi_d & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & -\phi_{d-1} & \cdots & 0 & 0 \\ 0 & -\phi_d & \cdots & 0 & 0 \end{pmatrix} \tag{3.6}$$

$$\boldsymbol{T_z} = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & \theta_1 & \cdots & \theta_{d-1} & \theta_d \\ 0 & \theta_2 & \cdots & \theta_d & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \theta_{d-1} & \cdots & 0 & 0 \\ 0 & \theta_d & \cdots & 0 & 0 \end{pmatrix} \tag{3.7}$$

Now it is necessary to address the two forms of the actual estimator and show where the vector in Equation 3.5 is used.

For the two-step Naive estimator, the first step is to take the data that is stored in the vector $\boldsymbol{z_n}$ and the past control inputs $\boldsymbol{u_n}$ and form the state vector needed in the state equation using Equation 3.5, which will perform in this case as the state estimator. The second step is to take the state vector and use it in the state equation to get the Naive estimate of state.

$$\begin{aligned} \boldsymbol{\hat{x}_{n+1|n}} &= \boldsymbol{A\hat{x}_{n|n}} + \boldsymbol{Bu_n} && (3.8) \\ &= \boldsymbol{A}\left(\boldsymbol{T_z z^*_n} + \boldsymbol{T_u u_n}\right) + \boldsymbol{Bu_n} && (3.9) \end{aligned}$$

This filter may be resistant to higher level robustness problems because of its pure time series setup. If there are parameter estimates that are poor due to lower level robustness problems, we only use them twice in the filter so we don't have as intense a recurring problem inside of one time step. The Kalman filter uses the $\boldsymbol{A}$ matrix at each successive step, compounding any error in its estimation. The Naive filter is very simple and is set up as the explicit structure defined in any filter. The setup of the equation implies this type of structure:

$$\boldsymbol{x_{n+1}} = \boldsymbol{Ax_n} + \boldsymbol{Bu_n} + \boldsymbol{G\epsilon_{n+1}}. \tag{3.10}$$

Here $\boldsymbol{x_{n+1}}$ is the state of the system at time $n+1$ estimated from the state at time n. It seems that this state estimate should have good properties due to such a simple yet direct connection to the theoretics of the system. The mathematical system is defined as

$$z_{n+1} = -\phi_1 z_n - \cdots - \phi_p z_{n-p} + \theta_1 u_n + \cdots + \theta_q u_q + \epsilon_{n+1}. \tag{3.11}$$

Because the estimate is set up with the same structure as the mathematical model, the estimate should be very good; however, notice that when the observation is used directly it shows a possible weakness to measurement error. We shall see in the simulations how this comes in to play in the actual evaluation. As said previously, the exponentially weigthted moving average (EWMA) will be an option to clean up much of the measurement error to boost the performance of the filter.

$$z^*_{n+1} = -\phi_1\left(z_n + \eta_n\right) - \cdots - \phi_p\left(z_{n-p} + \eta_{n-p}\right) + \theta_1 u_n + \cdots + \theta_q u_q + \epsilon_{n+1} + \eta_{n+1} \quad (3.12)$$

Because the observations are used directly, the measurement error has direct impact on the filter. As seen in (3.12), the measurement error saturates the process. Each time step the measurement error is present. This process error is a stochastic process that has a variance of roughly an order of magnitude larger than the typical process error variance. This really throws off some filters, especially those that do not have a method of smoothing out this measurement error. Earlier the Kalman filter was shown to adjust for measurement error and initial conditions, and specific focus was placed on Equation (4.2). This is an interesting property; the Kalman filter contains somewhat of an outlier cleaner. Since the measurement error is an additive outlier, the Kalman filter has some robustness properties to measurement error. The Naive filter does not have such an adjustment and could possibly suffer from this.

The EWMA discussed previously would be a wonderful candidate to add to the Naive filter. One would need to find the smoothing parameter for this smoothing operator first. One other possible way of making the Naive filter robust could be a bias correction. The bias of the Naive filter is simple. In the following equation, this bias is seen to be a function of the bias of the the time series parameters. This reveals a possible avenue for finding a bias correction. Bias is equivalent in this case to the generic uncertainty spoken of in Lee et al. (2004). Which has been derived in this case for the Naive filter and is shown in (3.13).

$$\hat{Bias}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n+1}}\right) = \begin{pmatrix} -\hat{Bias}\left(\hat{\phi}_1\right) z_n - \hat{Bias}\left(\hat{\phi}_2\right) z_{n-1} + \hat{Bias}\left(\hat{\theta}_1\right) u_n + \hat{Bias}\left(\hat{\theta}_2\right) u_{n-1} \\ -\hat{Bias}\left(\hat{\phi}_2\right) z_n + \hat{Bias}\left(\hat{\theta}_2\right) u_n \end{pmatrix} \quad (3.13)$$

This bias, along with the EWMA, will be applied to the Naive state estimate, to enhance the performance of the Naive filter.

4. DERIVATION OF THE NAIVE STATE VECTOR

Some processes represent only an autoregressive component, some a moving average, and there are many that exhibit a combination of the two. Processes with a combination, denoted as ARMA (autoregressive moving average), are of interest here. The only difference in the model dealt with here is that there is no moving average part but an exogenous input, termed the impulse response (IR). An outside force or influence adds an impulse to the system, which lasts in the system in memory described by the IR coefficients. If the autoregressive part were unstable or nonstationary, this model could be used to represent controlling inputs into the system to compensate. As mentioned, another possibility is that there are outside influences that affect the system independent of the process. We call these situations a dynamic system.

We will now give a description of a dynamic system in a two-step theoretical representation. The first step is a time equation that represents how the system is affected by time with the previous values of the exogenous inputs and states

$$\boldsymbol{x_n} = \boldsymbol{A}\boldsymbol{x_{n-1}} + \boldsymbol{B}\boldsymbol{u_{n-1}} + \boldsymbol{G}\boldsymbol{\epsilon_n}. \tag{4.1}$$

Here the matrix **A** describes the dynamics of the system, **B** is a matrix that describes the weights of the exogenous inputs, and **G** represents the system correlation. These matrices were described in more detail earlier in the literature review. The second step is a measurement equation for the dynamic system based on the current state of the system and the measurement error

$$\boldsymbol{y_n} = \boldsymbol{C}\boldsymbol{x_n} + \boldsymbol{\eta_n}. \tag{4.2}$$

The dynamic system has been defined as an autoregressive impulse response of order

(2,2) as follows:

$$z_n + \phi_1 z_{n-1} + \phi_2 z_{n-2} = \theta_1 u_{n-1} + \theta_2 u_{n-2} + \epsilon_n. \tag{4.3}$$

The inputs u stabilize the system if it is out of control. This has a linear filter describing the state of the system at time n as follows:

$$\boldsymbol{x_n} = \begin{pmatrix} z_n \\ -\phi_2 z_{n-1} + \theta_2 u_{n-1} \end{pmatrix} \tag{4.4}$$

$$\boldsymbol{A} = \begin{pmatrix} -\phi_1 & 1 \\ -\phi_2 & 0 \end{pmatrix} \tag{4.5}$$

$$\boldsymbol{B} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}. \tag{4.6}$$

Given the above representation, we can predict $\boldsymbol{x_n}$ given $\boldsymbol{x_{n-1}}$. But what is the general form for the matrices **A** and **B**? Also, what is the general form for the Naive filter? These questions are answered in the next few sections.

4.1 Preliminaries

We will derive the filter by solving for a generic **A** matrix and state vector. While deriving this filter, we will also establish the anchor of the proof of the general form. Hence we will now derive the relation for the (3,3) filter. We start by defining generic **A** and $\boldsymbol{x_{n-1}}$ matrices

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A_1} \\ \boldsymbol{A_2} \\ \boldsymbol{A_3} \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}$$

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B_1} \\ \boldsymbol{B_2} \\ \boldsymbol{B_3} \end{pmatrix} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix}.$$

Also, a generic state vectors for time steps n and *n-1* are represented as

$$\boldsymbol{x_n} = \begin{pmatrix} x_{n,1} \\ x_{n,2} \\ x_{n,3} \end{pmatrix}$$

$$\boldsymbol{x_{n-1}} = \begin{pmatrix} z_{n-1} \\ \delta_1 z_{n-2} + \delta_2 z_{n-3} + \omega_1 u_{n-2} + \omega_2 u_{n-3} \\ \eta_1 z_{n-2} + \eta_2 z_{n-3} + \beta_1 u_{n-2} + \beta_2 u_{n-3} \end{pmatrix}.$$

4.2 Derivation

We will now evaluate through the expression for the filter, which is

$$\boldsymbol{x_n = Ax_{n-1} + Bu_{n-1}}. \tag{4.7}$$

The first element of the left side of this equation is

$$x_{n,1} = -\phi_1 z_{n-1} - \phi_2 z_{n-2} - \phi_3 z_{n-3} + \theta_1 u_{n-1} + \theta_2 u_{n-2} + \theta_3 u_{n-3}. \tag{4.8}$$

The first element of the right side of the equation is $\boldsymbol{A_1 x_{n-1} + B_1 u_{n-1}}$ and is evaluated as

$$\begin{aligned} a_1 z_{n-1} + \\ b_1\delta_1 z_{n-2} + b_1\delta_2 z_{n-3} + b_1\omega_1 u_{n-2} + b_1\omega_2 u_{n-3} + \\ c_1\eta_1 z_{n-2} + c_1\eta_2 z_{n-3} + c_1\beta_1 u_{n-2} + c_1\beta_2 u_{n-3} + \\ \theta_1 u_{n-1}. \end{aligned}$$

Now we can rearrange the above expressions to reflect equalities to (4.8) by doing a little algebra.

z_{n-1}:

$$a_1 z_{n-1} = -\phi_1 z_{n-1} \quad \Rightarrow \quad a_1 = -\phi_1 \tag{4.9}$$

z_{n-2}:

$$(b_1\delta_1 + c_1\eta_1)\, z_{n-2} = -\phi_2 z_{n-2} \quad \Rightarrow \quad b_1\delta_1 + c_1\eta_1 = -\phi_2 \tag{4.10}$$

z_{n-3}:

$$(b_1\delta_2 + c_1\eta_2)\, z_{n-3} = -\phi_3 z_{n-3} \quad \Rightarrow \quad b_1\delta_2 + c_1\eta_2 = -\phi_3 \tag{4.11}$$

u_{n-2}:

$$(b_1\omega_1 + c_1\beta_1)\, u_{n-2} = \theta_2 u_{n-2} \quad \Rightarrow \quad b_1\omega_1 + c_1\beta_1 = \theta_2 \tag{4.12}$$

u_{n-3}:

$$(b_1\omega_2 + c_1\beta_2)\, u_{n-3} = \theta_3 u_{n-3} \quad \Rightarrow \quad b_1\omega_2 + c_1\beta_2 = \theta_3 \tag{4.13}$$

Now we move on to the next set of equations to solve this system. The next value we will consider from the state vector is $x_{n,2}$.

$$x_{n,2} = \delta_1 z_{n-1} + \delta_2 z_{n-2} + \omega_1 u_{n-1} + \omega_2 u_{n-2} \tag{4.14}$$

The second element of the right side of equation (4.7) is $\boldsymbol{A_2 x_{n-1} + B_2 u_{n-1}}$ and is evaluated as

$$\begin{aligned} a_2 z_{n-1} &+ \\ b_2\delta_1 z_{n-2} + b_2\delta_2 z_{n-3} + b_2\omega_1 u_{n-2} + b_2\omega_2 u_{n-3} &+ \\ c_2\eta_1 z_{n-2} + c_2\eta_2 z_{n-3} + c_2\beta_1 u_{n-2} + c_2\beta_2 u_{n-3} &+ \\ \theta_2 u_{n-1}&. \end{aligned}$$

Again rearranging these terms to resemble those in (4.14), we have

z_{n-1}:

$$a_2 z_{n-1} = \delta_1 z_{n-1} \quad \Rightarrow \quad a_2 = \delta_1 \tag{4.15}$$

z_{n-2}:

$$(b_2\delta_1 + c_2\eta_1)\, z_{n-2} = \delta_2 z_{n-2} \quad \Rightarrow \quad b_2\delta_1 + c_2\eta_1 = \delta_2 \tag{4.16}$$

z_{n-3}:

$$\left(b_2\delta_2 + c_2\eta_2\right) z_{n-3} = 0 \quad \Rightarrow \quad b_2\delta_2 + c_2\eta_2 = 0 \tag{4.17}$$

u_{n-1}:

$$\theta_2 u_{n-1} = \omega_1 u_{n-1} \quad \Rightarrow \quad \theta_2 = \omega_1 \tag{4.18}$$

u_{n-2}:

$$\left(b_2\omega_1 + c_2\beta_1\right) u_{n-2} = \omega_2 u_{n-2} \quad \Rightarrow \quad b_2\omega_1 + c_2\beta_1 = \omega_2 \tag{4.19}$$

u_{n-3}:

$$\left(b_2\omega_2 + c_2\beta_2\right) u_{n-3} = 0 \quad \Rightarrow \quad b_2\omega_2 + c_2\beta_2 = 0 \tag{4.20}$$

Here is the final set of equations we need from (4.7) expressed as $\boldsymbol{A_3 x_{n-1}} + \boldsymbol{B_3 u_{n-1}}$. These are compared to the equation (4.21) below.

$$\begin{aligned} a_3 z_{n-1} &+ \\ b_3\delta_1 z_{n-2} + b_3\delta_2 z_{n-3} + b_3\omega_1 u_{n-2} + b_3\omega_2 u_{n-3} &+ \\ c_3\eta_1 z_{n-2} + c_3\eta_2 z_{n-3} + c_3\beta_1 u_{n-2} + c_3\beta_2 u_{n-3} &+ \\ \theta_3 u_{n-1} & \end{aligned}$$

$$x_{n,3} = \eta_1 z_{n-1} + \eta_2 z_{n-2} + \beta_1 u_{n-1} + \beta_2 u_{n-2} \tag{4.21}$$

Rearrangement of the equation gives us the last set of equations needed to solve the (3,3) autoregressive impulse response dynamic system.

z_{n-1}:

$$a_3 z_{n-1} = \eta_1 z_{n-1} \quad \Rightarrow \quad a_3 = \eta_1 \tag{4.22}$$

z_{n-2}:

$$\left(b_3\delta_1 + c_3\eta_1\right) z_{n-2} = \eta_2 z_{n-2} \quad \Rightarrow \quad b_3\delta_1 + c_3\eta_1 = \eta_2 \tag{4.23}$$

z_{n-3}:

$$(b_3\delta_2 + c_3\eta_2)\, z_{n-3} = 0 \quad \Rightarrow \quad b_3\delta_2 + c_3\eta_2 = 0 \tag{4.24}$$

u_{n-1}:

$$\theta_3 u_{n-1} = \beta_1 u_{n-1} \quad \Rightarrow \quad \theta_3 = \beta_1 \tag{4.25}$$

u_{n-2}:

$$(b_3\omega_1 + c_3\beta_1)\, u_{n-2} = \beta_2 u_{n-2} \quad \Rightarrow \quad b_3\omega_1 + c_3\beta_1 = \beta_2 \tag{4.26}$$

u_{n-3}:

$$(b_3\omega_2 + c_3\beta_2)\, u_{n-3} = 0 \quad \Rightarrow \quad b_3\omega_2 + c_3\beta_2 = 0 \tag{4.27}$$

The next step is to put all of these expressions into a solution for each one of the parameters. We know by (4.18) and (4.25) that the following expression indicates what b_1 and c_1 are. Using (4.12) we find that $b_1 = 1$ and $c_1 = 0$ since

$$b_1\omega_1 + c_1\beta_1 = b_1\theta_2 + c_1\theta_3 = \theta_2.$$

With (4.13) we can show that $\omega_2 = \theta_3 = \beta_1$ by

$$b_1\omega_2 + c_1\beta_2 = (1)\omega_2 + (0)\beta_2 = \theta_3.$$

Since $a_2 = \delta_1$ and $a_3 = \eta_1$, (4.10) gives us

$$\begin{aligned} b_1\delta_1 + c_1\eta_1 &= (1)a_2 + (0)a_3 = -\phi_2 \\ a_2 &= \delta_1 = -\phi_2. \end{aligned}$$

For δ_2 we can use (4.11) to find a chain of equalities to show $a_3 = -\phi_3$.

$$\begin{aligned} b_1\delta_2 + c_1\eta_2 &= (1)\delta_2 + (0)\eta_2 = -\phi_3 \\ &\Rightarrow \quad \delta_2 = -\phi_3. \end{aligned}$$

Now we must find what b_2 and c_2 are. To do this we use (4.17) and note that $\delta_2 = -\phi_3$ as previously shown.

$$\begin{aligned} b_2\delta_2 + c_2\eta_2 &= b_2(-\phi_3) + c_2\eta_2 = 0 \\ \Rightarrow &\quad b_2 = 0 \end{aligned}$$

By (4.19) we find that $c_2 = 1$ since $\beta_1 = \omega_2$.

$$\begin{aligned} b_2\omega_1 + c_2\beta_1 &= b_2\omega_1 + c_2\omega_2 = \omega_2 \\ \Rightarrow &\quad c_2 = 1 \\ \Rightarrow &\quad \eta_2 = 0 \end{aligned}$$

We get η_2 from the equalities found before this, specifically from (4.19). Also, by (4.20) we know that $\beta_2 = 0$ since $c_2 = 1$ and $b_2 = 0$.

$$b_2\omega_2 + c_2\beta_2 = 0$$

We now find that $a_3 = -\phi_3$ by (4.16).

$$\begin{aligned} b_2\delta_1 + c_2\eta_1 = (0)\delta_1 + (1)\eta_1 = \delta_2 \\ \eta_1 = \delta_2 = a_3 \\ \textit{and} \\ \eta_1 = -\phi_3 \end{aligned}$$

Finally, $b_3 = c_3 = 0$ using the relation (4.23) and using the fact that $\delta_1 \neq 0$ and $\beta_1 \neq 0$ along with $\eta_2 = 0$.

$$b_3\delta_1 + c_3\eta_1 = \eta_2 \tag{4.28}$$

Having solved for all of the unknowns in this dynamic system, we can see the form of the matrices and vectors for the (2,2) and (3,3) cases. The setup for the (3,3) case of the Naive filter to estimate z_n is

$$\boldsymbol{A} = \begin{pmatrix} -\phi_1 & 1 & 0 \\ -\phi_2 & 0 & 1 \\ -\phi_3 & 0 & 0 \end{pmatrix} \tag{4.29}$$

$$\boldsymbol{x_{n-1}} = \begin{pmatrix} z_{n-1} \\ -\phi_2 z_{n-2} - \phi_3 z_{n-3} + \theta_2 u_{n-2} + \theta_3 u_{n-3} \\ -\phi_3 z_{n-2} + \theta_3 u_{n-2} \end{pmatrix}. \tag{4.30}$$

Validation of this solution can be found by showing that $\boldsymbol{x_n = Ax_{n-1} + Bu_{n-1} + G\epsilon_n}$ equates to the proper form.

$$\begin{aligned}
\boldsymbol{x_n} &= \boldsymbol{Ax_{n-1} + Bu_{n-1} + G\epsilon_n} \\
&= \begin{pmatrix} -\phi_1 & 1 & 0 \\ -\phi_2 & 0 & 1 \\ -\phi_3 & 0 & 0 \end{pmatrix} \begin{pmatrix} z_{n-1} \\ -\phi_2 z_{n-2} - \phi_3 z_{n-3} + \theta_2 u_{n-2} + \theta_3 u_{n-3} \\ -\phi_3 z_{n-2} + \theta_3 u_{n-2} \end{pmatrix} \\
&+ \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix} u_{n-1} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \epsilon_n \\ \epsilon_{n-1} \\ \epsilon_{n-2} \end{pmatrix} \\
&= \begin{pmatrix} -\phi_1 z_{n-1} - \phi_2 z_{n-2} - \phi_3 z_{n-3} + \theta_1 u_{n-1} + \theta_2 u_{n-2} + \theta_3 u_{n-3} + \epsilon_n \\ -\phi_2 z_{n-1} - \phi_3 z_{n-2} + \theta_2 u_{n-1} + \theta_3 u_{n-2} \\ -\phi_3 z_{n-1} + \theta_3 u_{n-1} \end{pmatrix} \\
&= \begin{pmatrix} z_n \\ -\phi_2 z_{n-1} - \phi_3 z_{n-2} + \theta_2 u_{n-1} + \theta_3 u_{n-2} \\ -\phi_3 z_{n-1} + \theta_3 u_{n-1} \end{pmatrix} = \boldsymbol{x_n}
\end{aligned} \tag{4.31}$$

This shows the form of the (3,3) filter and validates that the solution is correct. Now we will check the (4,4) case.

$$\begin{aligned}
\boldsymbol{x_n} &= \begin{pmatrix} -\phi_1 & 1 & 0 & 0 \\ -\phi_2 & 0 & 1 & 0 \\ -\phi_3 & 0 & 0 & 1 \\ -\phi_4 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} z_{n-1} \\ -\phi_2 z_{n-2} - \phi_3 z_{n-3} - \phi_4 z_{n-4} + \theta_2 u_{n-2} + \theta_3 u_{n-3} + \theta_4 u_{n-4} \\ -\phi_3 z_{n-2} - \phi_4 z_{n-3} + \theta_3 u_{n-2} + \theta_4 u_{n-3} \\ -\phi_4 z_{n-2} + \theta_4 u_{n-2} \end{pmatrix} \\
&+ \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{pmatrix} u_{n-1} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \epsilon_n \\ \epsilon_{n-1} \\ \epsilon_{n-2} \\ \epsilon_{n-3} \end{pmatrix} \\
&= \begin{pmatrix} -\phi_1 z_{n-1} - \phi_2 z_{n-2} - \phi_3 z_{n-3} - \phi_4 z_{n-4} + \theta_1 u_{n-1} + \theta_2 u_{n-2} + \theta_3 u_{n-3} + \theta_4 u_{n-4} + \epsilon_n \\ -\phi_2 z_{n-1} - \phi_3 z_{n-2} - \phi_4 z_{n-3} + \theta_2 u_{n-1} + \theta_3 u_{n-2} + \theta_4 u_{n-3} \\ -\phi_3 z_{n-1} - \phi_4 z_{n-2} + \theta_3 u_{n-1} + \theta_4 u_{n-2} \\ -\phi_4 z_{n-1} + \theta_4 u_{n-1} \end{pmatrix} \\
&= \begin{pmatrix} z_n \\ -\phi_2 z_{n-1} - \phi_3 z_{n-2} - \phi_4 z_{n-3} + \theta_2 u_{n-1} + \theta_3 u_{n-2} + \theta_4 u_{n-3} \\ -\phi_3 z_{n-1} - \phi_4 z_{n-2} + \theta_3 u_{n-1} + \theta_4 u_{n-2} \\ -\phi_4 z_{n-1} + \theta_4 u_{n-1} \end{pmatrix} \qquad (4.32)
\end{aligned}$$

This shows that we can generalize the form as a (p,q) autoregressive impulse response Naive filter.

4.3 General Form of the Naive Filter

Now we can use some notation to generalize the notation of this filter, and note that whether $p = q,\ p > q$, or $p < q$, we can still use the same general form. We start with the special case, $p = q$. The general form is given as

$$\boldsymbol{A} = \begin{pmatrix} -\phi_1 & 1 & 0 & \cdots & 0 \\ -\phi_2 & 0 & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ -\phi_{p-1} & 0 & 0 & & 1 \\ -\phi_p & 0 & 0 & \cdots & 0 \end{pmatrix}_{p\times p} \tag{4.33}$$

$$= \begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{I_0} \end{bmatrix}. \tag{4.34}$$

Above in (4.34) we are creating some notation that will be helpful in the derivation of this special case of the Naive filter. We denote $\boldsymbol{I_0}$ as a $(p-1)\times(p-1)$ identity with a zero $p-1$ dimension row vector appended to the bottom as such.

$$\boldsymbol{I_0} = \begin{pmatrix} \boldsymbol{I_{(p-1)\times(p-1)}} \\ \boldsymbol{0_{1\times(p-1)}} \end{pmatrix}$$

The state vector has some interesting structure dealing with index shifts in summations. This vector will also have a definition of special notation to make the derivation more tractable. We start with a general definition for the j^{th} element of $\boldsymbol{x_n}$. Note that this defines the value z_n as well, since in theory a summation of all the p previous z values and inputs make up this new measure with the addition of the process error for this step.

$$\boldsymbol{x_n} = \left\{ \sum_{i=j}^{p} -\phi_i z_{n-i+(j-1)} + \theta_i u_{n-i+(j-1)} \right\}_j \tag{4.35}$$

$$= \begin{pmatrix} z_n \\ \sum_{i=2}^{p} -\phi_i z_{n-i+1} + \theta_i u_{n-i+1} \\ \sum_{i=3}^{p} -\phi_i z_{n-i+2} + \theta_i u_{n-i+2} \\ \sum_{i=4}^{p} -\phi_i z_{n-i+3} + \theta_i u_{n-i+3} \\ \vdots \\ \sum_{i=p-1}^{p} -\phi_i z_{n-i+(p-2)} + \theta_i u_{n-i+(p-2)} \\ \sum_{i=p}^{p} -\phi_i z_{n-i+(p-1)} + \theta_i u_{n-i+(p-1)} \end{pmatrix}_{p\times 1}$$

$$= \begin{pmatrix} z_n \\ \boldsymbol{s}_n \end{pmatrix} \tag{4.36}$$

The notation in (4.36) is defined as the first element of the state vector, z_n, and the rest of the values which are made up of previous values of the process. The vector $\boldsymbol{s_n}$ represents all of the summations in (4.35), basically the elements 2 through p of $\boldsymbol{x_n}$. So, the dimension of $\boldsymbol{s_n}$ is $(p-1) \times 1$.

For the remainder of the definition we have

$$\boldsymbol{B} = \begin{pmatrix} \theta_1 & \theta_2 & \cdots & \theta_{p-1} & \theta_p \end{pmatrix}^T_{p\times 1} \tag{4.37}$$

$$\boldsymbol{u_{n-1}} = (u_{n-1})_{1\times 1} \tag{4.38}$$

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}_{p\times p} \tag{4.39}$$

$$\boldsymbol{\epsilon_n} = \begin{pmatrix} \epsilon_n & \epsilon_{n-1} & \cdots & \epsilon_{n-p+1} & \epsilon_{n-p} \end{pmatrix}^T_{p\times 1}. \tag{4.40}$$

4.4 Proof of the General Naive Filter

The general form of the Naive filter has been proposed but we don't know for sure if it truly gives the correct form for any number $p = q$. Because of this, we will use a simple direct proof to show that this filter will give us the desired state for any p. It has already been shown what the state should be for the n^{th} case now to show that it is the same for the $(n+1)^{th}$ case.

Assume now that the filter for the n^{th} case is of the form

$$\boldsymbol{x_n} = \boldsymbol{A x_{n-1}} + \boldsymbol{B u_{n-1}} + \boldsymbol{G \epsilon_n}$$

and described in detail in (4.35) above. This is what the form is for this case, but for the $(n+1)^{th}$ case we would expect to see the following:

$$\boldsymbol{x_n} = \begin{pmatrix} z_n \\ \sum_{i=2}^{p} -\phi_i z_{n-i+1} + \theta_i u_{n-i+1} \\ \sum_{i=3}^{p} -\phi_i z_{n-i+2} + \theta_i u_{n-i+2} \\ \sum_{i=4}^{p} -\phi_i z_{n-i+3} + \theta_i u_{n-i+3} \\ \vdots \\ \sum_{i=p-1}^{p} -\phi_i z_{n-i+(p-2)} + \theta_i u_{n-i+(p-2)} \\ \sum_{i=p}^{p} -\phi_i z_{n-i+(p-1)} + \theta_i u_{n-i+(p-1)} \end{pmatrix}. \tag{4.41}$$

Next it is necessary to show what the form is for the next case, n + 1. We use the relation $\boldsymbol{x_{n+1} = Ax_n + Bu_n + G\epsilon_{n+1}}$ to show this. Here we already have the expression for $\boldsymbol{x_n}$ and assume it is true. Having done this, is $\boldsymbol{x_{n+1}}$ of the same form? It will now be shown that indeed it is. To give a clear view of what is going on, we will use the notation introduced previously in (4.34) and (4.36).

$$\begin{aligned} \boldsymbol{x_{n+1}} &= \boldsymbol{Ax_n + Bu_n + G\epsilon_{n+1}} = \boldsymbol{\Phi} z_n + \boldsymbol{I_0 s_n} + \boldsymbol{Bu_n} + \boldsymbol{G\epsilon_{n+1}} \qquad (4.42) \\ &= \begin{pmatrix} -\phi_1 z_n \\ -\phi_2 z_n \\ \vdots \\ -\phi_{p-1} z_n \\ -\phi_p z_n \end{pmatrix} + \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \begin{pmatrix} \sum_{i=2}^{p} -\phi_i z_{n-i+1} + \theta_i u_{n-i+1} \\ \sum_{i=3}^{p} -\phi_i z_{n-i+2} + \theta_i u_{n-i+2} \\ \sum_{i=4}^{p} -\phi_i z_{n-i+3} + \theta_i u_{n-i+3} \\ \vdots \\ \sum_{i=p-1}^{p} -\phi_i z_{n-i+(p-2)} + \theta_i u_{n-i+(p-2)} \\ \sum_{i=p}^{p} -\phi_i z_{n-i+(p-1)} + \theta_i u_{n-i+(p-1)} \end{pmatrix} \\ &+ \begin{pmatrix} \theta_1 u_n \\ \theta_2 u_n \\ \vdots \\ \theta_{p-1} u_n \\ \theta_p u_n \end{pmatrix} + \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \begin{pmatrix} \epsilon_{n+1} \\ \epsilon_n \\ \vdots \\ \epsilon_{n-p+2} \\ \epsilon_{n-p+1} \end{pmatrix} \end{aligned}$$

$$= \begin{pmatrix} -\phi_1 z_n \\ -\phi_2 z_n \\ \vdots \\ -\phi_{p-1} z_n \\ -\phi_p z_n \end{pmatrix} + \begin{pmatrix} \sum_{i=2}^{p} -\phi_i z_{n-i+1} + \theta_i u_{n-i+1} \\ \sum_{i=3}^{p} -\phi_i z_{n-i+2} + \theta_i u_{n-i+2} \\ \sum_{i=4}^{p} -\phi_i z_{n-i+3} + \theta_i u_{n-i+3} \\ \vdots \\ \sum_{i=p-1}^{p} -\phi_i z_{n-i+(p-2)} + \theta_i u_{n-i+(p-2)} \\ \sum_{i=p}^{p} -\phi_i z_{n-i+(p-1)} + \theta_i u_{n-i+(p-1)} \\ 0 \end{pmatrix}$$

$$+ \begin{pmatrix} \theta_1 u_n \\ \theta_2 u_n \\ \vdots \\ \theta_{p-1} u_n \\ \theta_p u_n \end{pmatrix} + \begin{pmatrix} \epsilon_{n+1} \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} -\phi_1 z_n - \phi_2 z_{n-1} - \cdots - \phi_p z_{n-p+1} + \theta_2 u_{n-1} + \cdots + \theta_p u_{n-p+1} \\ -\phi_2 z_n - \phi_3 z_{n-1} - \cdots - \phi_p z_{n-p+2} + \theta_3 u_{n-1} + \cdots + \theta_p u_{n-p+2} \\ \vdots \\ -\phi_{p-1} z_n - \phi_p z_{n-1} + \theta_p u_{n-1} \\ -\phi_p z_n \end{pmatrix}$$

$$+ \begin{pmatrix} \theta_1 u_n \\ \theta_2 u_n \\ \vdots \\ \theta_{p-1} u_n \\ \theta_p u_n \end{pmatrix} + \begin{pmatrix} \epsilon_{n+1} \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} -\phi_1 z_n - \phi_2 z_{n-1} - \cdots - \phi_p z_{n-p+1} + \theta_1 u_n + \theta_2 u_{n-1} + \cdots + \theta_p u_{n-p+1} + \epsilon_{n+1} \\ -\phi_2 z_n - \phi_3 z_{n-1} - \cdots - \phi_p z_{n-p+2} + \theta_2 u_n + \theta_3 u_{n-1} + \cdots + \theta_p u_{n-p+2} \\ \vdots \\ -\phi_{p-1} z_n - \phi_p z_{n-1} + \theta_{p-1} u_n + \theta_p u_{n-1} \\ -\phi_p z_n + \theta_p u_n \end{pmatrix}$$

$$
\begin{aligned}
&= \begin{pmatrix} z_{n+1} \\ \sum_{i=2}^{p} -\phi_i z_{n-i+2} + \theta_i u_{n-i+2} \\ \sum_{i=3}^{p} -\phi_i z_{n-i+3} + \theta_i u_{n-i+3} \\ \sum_{i=4}^{p} -\phi_i z_{n-i+4} + \theta_i u_{n-i+4} \\ \vdots \\ \sum_{i=p-1}^{p} -\phi_i z_{n-i+(p-1)} + \theta_i u_{n-i+(p-1)} \\ \sum_{i=p}^{p} -\phi_i z_{n-i+(p)} + \theta_i u_{n-i+(p)} \end{pmatrix} \\
&= \boldsymbol{x_{n+1}}
\end{aligned}
$$

This state vector for the $(n+1)^{th}$ case is the same as that found in (4.35), with the only difference being the change in the indices. We now know that the relation we have seen previously in (4.33-4.40) is true for any n. ***QED***

4.5 Cases $p > q$ and $p < q$

This section shows how the filter is used when $p \neq q$. In this case we just set some of the parameters equal to zero. We will consider the first case, a dynamic system of (4,2) order. Hence there are four autoregressive parameters and two impulse response parameters. The $\boldsymbol{A}$ matrix and the vector $\boldsymbol{B}$ are defined in the following way after the general form

$$
\begin{aligned}
\boldsymbol{A} &= \begin{pmatrix} -\phi_1 & 1 & 0 & 0 \\ -\phi_2 & 0 & 1 & 0 \\ -\phi_3 & 0 & 0 & 1 \\ -\phi_4 & 0 & 0 & 0 \end{pmatrix} \\
\boldsymbol{B} &= \begin{pmatrix} \theta_1 \\ \theta_2 \\ 0 \\ 0 \end{pmatrix}.
\end{aligned}
$$

We end up with a state vector that looks exactly the same, except that when we sum across some of the indices for the θs we just sum with these equated to zero. Thus, after evaluating $\boldsymbol{x_n = Ax_{n-1} + Bu_{n-1} + G\epsilon_n}$, the state vector looks like this,

$$\boldsymbol{x_n} = \begin{pmatrix} -\phi_1 z_{n-1} - \phi_2 z_{n-2} - \phi_3 z_{n-3} - \phi_4 z_{n-4} + \theta_1 u_{n-1} + \theta_2 u_{n-2} + \epsilon_n \\ -\phi_2 z_{n-1} - \phi_3 z_{n-2} - \phi_4 z_{n-3} + \theta_2 u_{n-1} \\ -\phi_3 z_{n-1} - \phi_4 z_{n-2} \\ -\phi_4 z_{n-1} \end{pmatrix}.$$

The form of the state vector for the $(n-1)^{th}$ time step looks just the same. We just set those specific parameters equal to zero if the order of the system dictates. Now if we have a (2,4) system we will get matrices as follows:

$$\begin{aligned} \boldsymbol{A} &= \begin{pmatrix} -\phi_1 & 1 & 0 & 0 \\ -\phi_2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \\ \boldsymbol{B} &= \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{pmatrix} \end{aligned}$$

where

$$\boldsymbol{x_{n-1}} = \begin{pmatrix} -\phi_1 z_{n-2} - \phi_2 z_{n-3} + \theta_1 u_{n-2} + \theta_2 u_{n-3} + \theta_3 u_{n-4} + \theta_4 u_{n-5} + \epsilon_{n-1} \\ -\phi_2 z_{n-1} + \theta_2 u_{n-2} + \theta_3 u_{n-3} + \theta_4 u_{n-4} \\ \theta_3 u_{n-2} + \theta_4 u_{n-3} \\ \theta_4 u_{n-2} \end{pmatrix}$$

5. ADJUSTED NAIVE FILTERS

The Naive filter itself is very susceptible to measurement error. A smoother of some sort is necessary to clear the effectual mud and allow the Naive filter to perform properly. In Chapter 3 there was a description of the bias of the Naive filter. This bias can be taken into account by correcting the Naive filter with the negative of the bias.

This correction takes the same form as the Naive filter. It is applied after the Naive filter has passed over the data. Given the estimate of state, $\hat{\boldsymbol{x}}_{\boldsymbol{n+1}}$, the correction is applied as

$$\hat{\boldsymbol{x}}_{\boldsymbol{n+1,BC}} = \boldsymbol{A}_{\boldsymbol{bias}}\hat{\boldsymbol{x}}_{\boldsymbol{n+1}} + \boldsymbol{B}_{\boldsymbol{bias}}\boldsymbol{u}_{\boldsymbol{n}}, \tag{5.1}$$

where

$$\boldsymbol{A}_{\boldsymbol{bias}} = \begin{bmatrix} \hat{Bias}\left(\hat{\phi}_1\right) & 1 \\ \hat{Bias}\left(\hat{\phi}_2\right) & 0 \end{bmatrix}, \tag{5.2}$$

$$\boldsymbol{B}_{\boldsymbol{bias}} = \begin{bmatrix} \hat{Bias}\left(\hat{\theta}_1\right) \\ \hat{Bias}\left(\hat{\theta}_2\right) \end{bmatrix}. \tag{5.3}$$

Basically, this correction is for the bias in the parameter estimate as seen in the equations. Bias here is the deviation of the parameter estimate from its true value. Another adjustment can be used to make the biggest difference in the performance of the Naive filter. The exponentially weighted moving average (EWMA) provides an excellent way of removing some of the measurement error from the measured value y_n. This smoother is established on an infinite linear relationship, but can be boiled down to a recursive equation. A smoothing parameter, λ_s, is chosen to start the process. The last estimated value from the EWMA is recorded for use in the next recursion.

$$\hat{z}_n = \lambda_s y_n + (1 - \lambda_s)\,\hat{z}_{n-1} \tag{5.4}$$

It is important to use the previous smoothed value of the EWMA estimate, not the previously estimated value from the Naive filter, because this does not work. This value is then fed into the Naive filter for estimation of state. The Bias Corrected EWMA Naive filter smoothes the next measured value with the EWMA, then the smoothed value is fed into the filter. After the estimate is complete in the Naive filter, the bias correction is made.

6. MSE SIMULATION

The goal in this thesis is to simulate a time series process with measurement error and try to estimate the state of the process in the presence of outliers and poor parameter estimates. Some of the filters considered have the ability to take knowledge of the uncertainties along with estimates of those parameters to robustly estimate the state of the process. As shown below, the time series process that will be used is an ARIR(2,2).

$$z_n = -\phi_1 z_{n-1} - \phi_2 z_{n-2} + \theta_1 u_{n-1} + \theta_2 u_{n-2} + \epsilon_n \tag{6.1}$$

$$y_n = z_n + \eta_n \tag{6.2}$$

The length of the trace or process that will be simulated will be done with two lengths, one at 12 and the other at 52. This is to inject an outlier in the process at times 10 and 50 and allow the filters two additional time steps to see the performance as the outlier passes out of the system. The state estimate is used normally in combination with a controller such as the LQR. Then the u_n would be control inputs to stabilize a system in a dynamic feedback loop. In this thesis the u_n will be considered as random exogenous inputs, essentially the reverse of control inputs. Following are definitions of all the stochastic components of the time series process.

$$\begin{aligned}
\epsilon_n &\sim N\left(0, \sigma_\epsilon^2\right) \\
\eta_n &\sim N\left(0, \sigma_\eta^2\right) \\
u_n &\sim N\left(0, \sigma_u^2\right) \\
a_{10} &\sim N\left(0, 5^2\sigma_u^2\right) \\
a_{50} &\sim N\left(0, 5^2\sigma_u^2\right) \\
z_{-1} &\sim N\left(0, \sigma_{pre}^2\right)
\end{aligned}$$

$$\begin{aligned} z_0 &\sim N\left(0, \sigma_{pre}^2\right) \\ \psi &\sim N\left(0, \sigma_{\psi}^2\right) \end{aligned}$$

In this array of stochastic parameters, certain of them are specific to this simulation. The random variables a_{10} and a_{50} are random outliers that come from a Gaussian with standard deviation five times greater than that of the exogenous input standard deviations. They are added in as denoted in the subscript two time steps before the end of the trace. For the trace of length 12 the outlier will be added into the input stream at time 10, and likewise for the trace of length 52. To start the simulation, instead of padding the beginning of the process with two zeros to feed into the process equations (6.1 & 6.2), some small white noise is put in for values z_{-1} and z_0.

As stated previously, some of the filters require an uncertain knowledge of the deviation of the parameter estimates from their true value. To model this uncertainty, ψ is used as this perturbation of the knowledge of the deviations. In the state equations given previously in the literature review, such as equation (2.3), the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are used directly. To simulate poor estimation of the parameters in these matrices, deviations are added to them. The deviations are $\delta_{\phi_1}, \delta_{\phi_2}, \delta_{\theta_1}, \delta_{\theta_2}$, one set for each parameter. They take on values $\{-0.5, 0, 0.5\}$, for which the simulated parameter estimate is set for investigative purposes as $\hat{\phi} = \phi + \delta_{\phi}$.

The filters that account for uncertainty in the parameter estimates will require the perturbations on the deviations. Exact knowledge of the uncertainty would assume we know what δ is for each parameter. To simulate an estimation procedure for the deviations, we let $\hat{\delta} = \delta + \psi$. This mimics taking the deviations from sampling distributions. In this case, the deviation from knowledge is set as a stochastic value that is randomly generated in ψ as given above in the definition of all stochastic values. The model parameters and standard deviations for the simulations are given below.

$$\begin{aligned}\phi_1 &= -0.95\\ \phi_2 &= 0.6\\ \theta_1 &= 0.6\\ \theta_2 &= 0.9\\ \sigma_\epsilon &= 1\\ \sigma_\eta &= 10\\ \sigma_u &= 3\\ \sigma_{pre} &= 0.1\\ \sigma_\psi &= 0.3\end{aligned}$$

When the trace for either trace size, 12 or 52, is generated using all these values and the equations above, the filters are used to estimate the state z_n using y_n. Let the variable $f(y_n)$ be a generic filter representing any of the filters compared in this thesis. After this estimate is found using a filter, the MSE is computed using the following equation.

$$MSE_f = \frac{1}{N-1}\sum_{n=1}^{N}\left(f(y_n) - z_n\right)^2 \tag{6.3}$$

In this equation N represents the trace size, either 12 or 52. Each deviation combination is used 100 times—that is, a combination of the deviations, δs, is used repetitively for 100 iterations. This is to generate the distribution of the MSEs for each combination. These distributions are saved on a CD placed in the back of this thesis. The presentation of these MSE distributions using box plots can also be found on the CD, as well as in appendix A.

By taking these MSEs and defining factor variables with three levels to represent the different deviations, eight levels to represent the filter type, and two levels to represent the size of the trace, we can model the effects of the deviations, filter type, and size of the trace on the value of MSE. An ANOVA model using a coding scheme of choice can be used to find the best model and to describe its properties.

7. RESULTS

Before fitting a model or looking at any analysis on the filters, it is important to realize that comparison of the filters overall and by trace size are the two most important considerations. The investigation into how the uncertainty affects the performance positively or negatively is of much less importance since there is no way of controlling or knowing exactly how inaccurate the estimate of uncertainty is at any given instance or situation. It is only meaningful to choose a filter knowing that it may be robust to a certain type of uncertainty, or to multiple types. If it is a good filter in a more general sense, it is due to a greater number of positive properties that work against uncertainty in parameter estimates. Investigation into the uncertainty effects will only be for the purpose of defining why the filter turned out to be ranked high in robustness.

The deviations range across three levels, $\{-0.5, 0, 0.5\}$. There are eight filter types, listed in order and with abbreviated names for plotting. They are

1. **(K1)** Krein space Kalman filter
2. **(K2)** Krein space Kalman filter with IR part added
3. **(K3)** Krein space Kalman filter with IR part and a second uncertainty parameter adjustment
4. **(K4)** Classical Kalman filter
5. **(N)** Naive filter
6. **(NBC)** Bias Corrected Naive filter
7. **(NE)** EWMA Naive filter

8. **(NEBC)** Bias Corrected EWMA Naive filter

As mentioned previously, there are only two sizes for the simulated traces, 12 and 52. The model that will be fit for the cell means model is

$$y_{hijk\ell mn} = \mu_{hijk\ell m} + \epsilon_{hijk\ell mn}. \tag{7.1}$$

The cell means model provides what is needed to test any hypothesis of interest here. The questions to answer are

1. Which filter is best?

2. How do the best three rank?

Using the cell means model and the `biglm()` function, the estimates of the cell means will provide a straightforward way of estimating and comparing the different filter behavior. The behavior is exhibited using the marginal means $\mu_{....\ell.}$ for $\ell = 1, \ldots, 8$. In Table 7.1 and Figure 7.1 there is no question as to whether the different filters are distinct as far as their performance. This enables the general ranking of the filters from best to poorest in estimating the state of the process across all situations presented by the deviations to the parameter estimates.

In this table, the designations of $Krein\ Kalman_2$ and $Krein\ Kalman_3$ represent the adjustments made to the Krein space Kalman filter presented by Lee et al. (2004). The first is the filter with the exogenous input adjustment, the second has the exogenous input and an additional accounting of the uncertainty parameter.

7.1 Overall Filter Comparison

These comparisons are enhanced by a graphical comparison in Figure 7.1. The four best filters by overall MSE are the $Krein\ Kalman_2$, $Krein\ Kalman_3$, *EWMA Naive*, and the *EWMA B. C. Naive* (B. C. stands for bias corrected). To enhance this comparison, these four filters will be shown in a graph together without the other

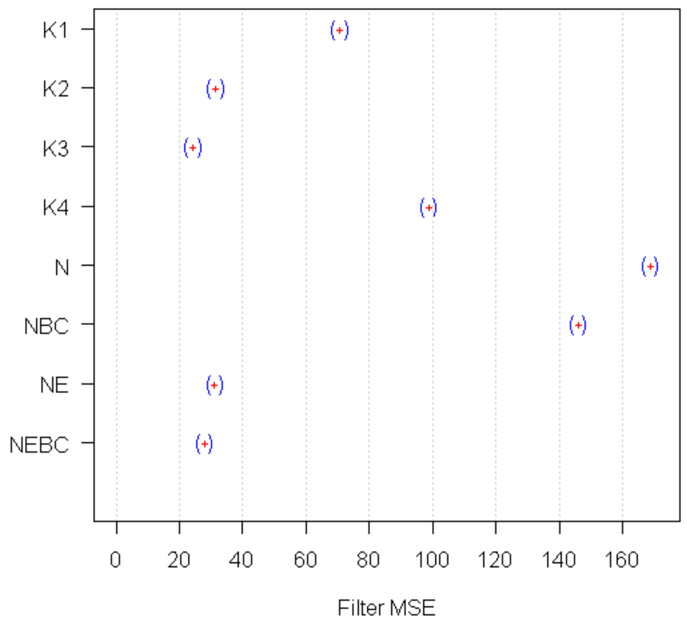


Figure 7.1: MSE 95% confidence intervals for MSE values per filter.

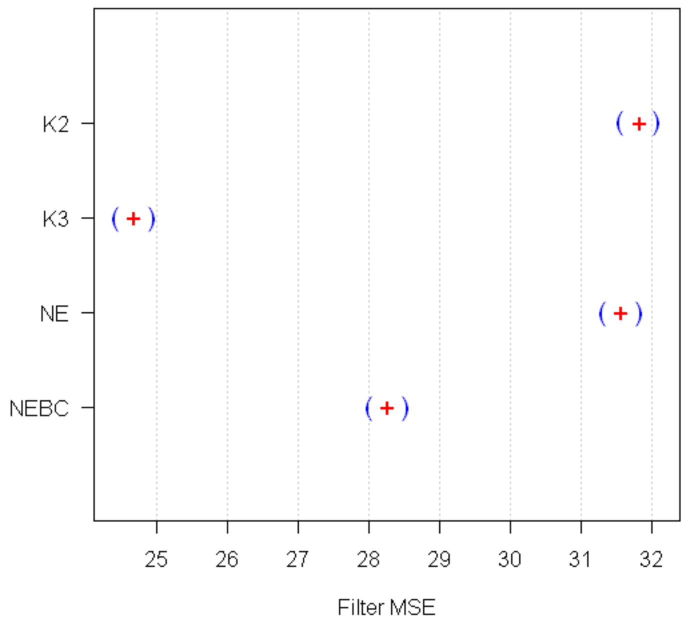


Figure 7.2: MSE 95% confidence intervals for MSE values per filter.

		MSE 95% CI		
	Filter	Lower	Estimate	Upper
K1	*Krein Kalman*	70.731	70.986	71.242
K2	*Krein Kalman*$_2$	31.569	31.824	32.079
K3	*Krein Kalman*$_3$	24.427	24.682	24.937
K4	*Classic Kalman*	98.892	99.148	99.403
N	*Naive*	168.825	169.080	169.335
NBC	*B. C. Naive*	145.966	146.221	146.476
NE	*EWMA Naive*	31.313	31.568	31.824
NEBC	*EWMA B. C. Naive*	28.011	28.267	28.522

Table 7.1: Comparison of the filter marginal mean MSE.

filters in Figure 7.2. In the graphs, it appears there could be no significant difference between the NE and K2 filters. A test of hypothesis confirms this. The individual tests of these differences provides a p-value of 0.1646 on this difference. It was decided upon to compare NE with K2, NE with NEBC, and NEBC with K3 because these comparisons seem to offer the smallest distance to one another. All but the first comparison prove to differ with p-values of 0. The top three filters consist of K3 as the best overall filter, NEBC as the second best filter, and NE and K2 as third best.

7.2 Filter Comparison by Trace Size

The same rankings can be made for the filters by looking at the estimates and confidence intervals by trace size.

Notice that with the smaller trace size, Figure 7.3, the Kalman filter becomes equal with the regular Krein Kalman filter. The dotted black lines in Figure 7.3 show the movement of each marginal mean for filter from the overall marginal mean, $\mu_{....\ell.}$, to the marginal mean accounting for trace size, $\mu_{....\ell 1}$, where the subscript 1 indicates it is a mean for a trace size of 12. It is interesting to note that the Classic Kalman, Naive, and Bias Corrected Naive filters actually drop in MSE mean value

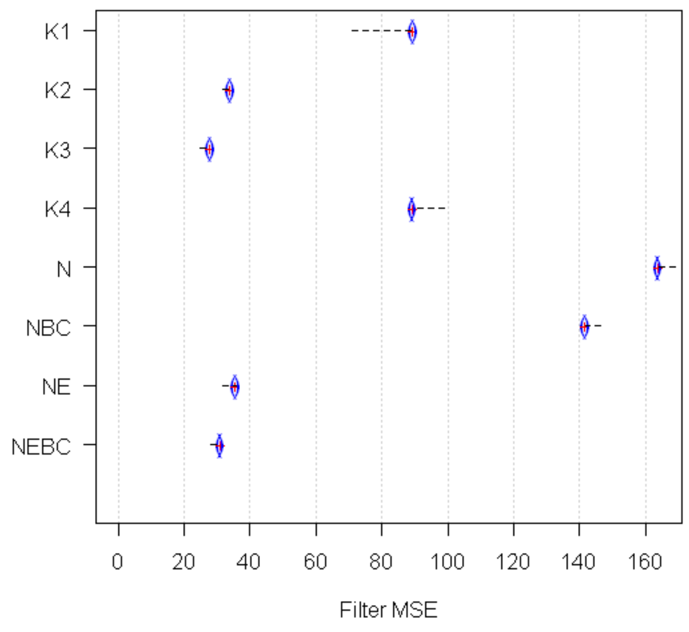


Figure 7.3: MSE 95% confidence intervals per filter simulated trace of size 12.

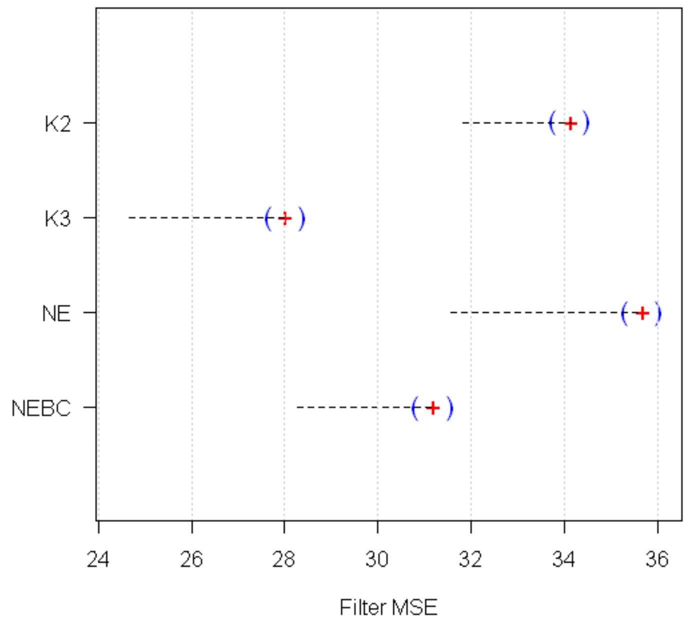


Figure 7.4: MSE 95% confidence intervals per selected filters simulated trace of size 12.

		MSE 95% CI		
	Filter	Lower	Estimate	Upper
K1	*Krein Kalman*	89.167	89.528	89.889
K2	*Krein Kalman*$_2$	33.770	34.131	34.492
K3	*Krein Kalman*$_3$	27.669	28.030	28.391
K4	*Classic Kalman*	89.129	89.490	89.851
N	*Naive*	163.610	163.970	164.331
NBC	*B. C. Naive*	141.458	141.819	142.180
NE	*EWMA Naive*	35.320	35.681	36.041
NEBC	*EWMA B. C. Naive*	30.835	31.196	31.560

Table 7.2: Comparison of the filter marginal mean MSE, (trace size of 12).

for the smaller trace size. The Kalman filter has an MSE value less than the overall value calculated over both trace sizes. The Krein space Kalman filter has an increase in MSE. This indicates a good property in the Classic Kalman filter; despite the large MSE value, it has a downward trend in MSE, making it essentially equal in performance with the Krein Kalman filter.

In Figure 7.4 the four highest ranked filters increase in MSE with a smaller trace size. They have separated somewhat in MSE value as well, the dotted black line recording the change in overall marginal filter MSE values, $\mu_{\dots.\ell.}$, to the trace size and filter marginal mean, $\mu_{\dots.\ell 2}$, for the trace of size 52. An individual test indicates they are all different, with the largest p-value of 2.688×10^{-9}. The confidence intervals agree with the hypotheses.

Ranking in this case breaks the tie between NE and K2. The MSE values have increased by breaking the comparisons by trace size. These four filter types have a tendency to have lower MSE in general, but their best performance is after the trace/signal has proceeded beyond this level into the infinite time horizon. The EWMA recursion that enhances the Naive filter is founded on an infinite time horizon principle and may explain some of the reason the two Naive filters lack slightly when the time index is close to the start time.

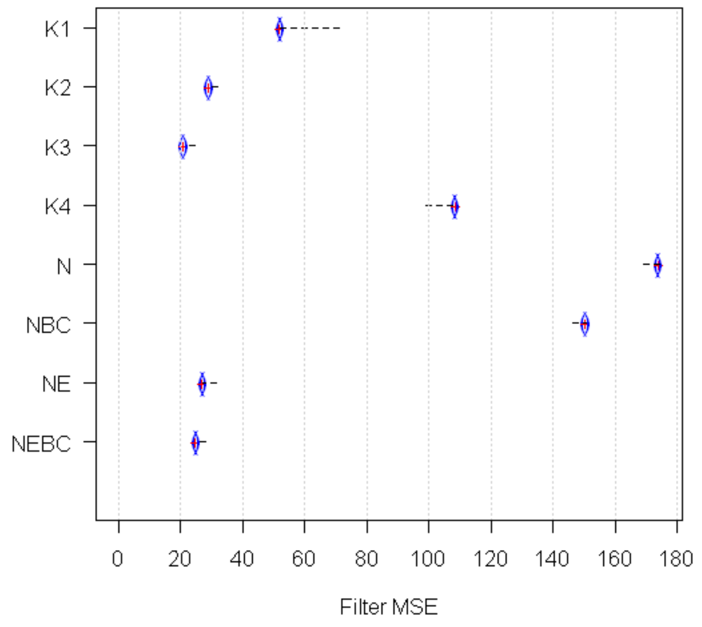


Figure 7.5: MSE 95% confidence intervals per filter simulated trace of size 52.

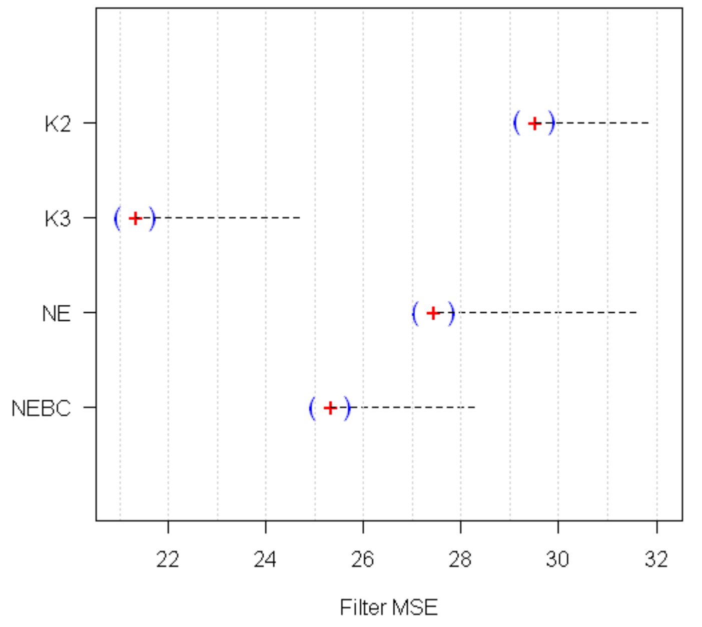


Figure 7.6: MSE 95% confidence intervals per selected filters simulated trace of size 52.

		MSE 95% CI		
	Filter	Lower	Estimate	Upper
K1	*Krein Kalman*	52.084	52.445	52.805
K2	*Krein Kalman*$_2$	29.157	29.518	29.878
K3	*Krein Kalman*$_3$	20.973	21.334	21.695
K4	*Classic Kalman*	108.444	108.805	109.166
N	*Naive*	173.829	174.190	174.551
NBC	*B. C. Naive*	150.262	150.623	150.983
NE	*EWMA Naive*	27.095	27.456	27.817
NEBC	*EWMA B. C. Naive*	24.977	25.338	25.698

Table 7.3: Comparison of the filter marginal mean MSE, (trace size of 52).

Now with the larger trace, there is an opposite effect. The Krein and Classic Kalman filters differ in performance. The Krein filter begins to approach peak performance; the Classic Kalman filter converges on its normal performance. The other filters seem to follow in this performance convergence. Those that decreased in MSE for a trace size of 12 are increasing for a trace size of 52, and vice versa.

The top two filters remain in first and second place, and the NE and K2 filters switch places. Just by looking at the confidence intervals it is very apparent that all four are very distinct. It can be deduced that for very short time traces when one expects the prediction interval to be very short, the third and fourth best alternatives are K2 and NE, respectively. When the case is the long term, or at least longer than around 52 samplings in time, the rank of these two filters reverses.

When considering the Krein space Kalman filter with exogenous input adjustment and secondary uncertainty enhancement and the Bias Corrected EWMA Naive filter, the clear winner is the enhanced Krein filter. This provides a nice property, that both are quite consistent under different conditions. The most inconsistent filter when trace size increases is the regular Krein space Kalman filter. In the graphs in this section, specifically Figures 7.3 and 7.5, it is quite obvious that the MSE values for this filter cover a far greater range than do any of the other filters, especially in comparison to the four best filters. It ranges from 89.528 for a trace of size 12 to 52.445 for a trace of 52 points. This is a larger range than all the other filters compared.

7.3 Interaction Interpretation

For thinking in terms of interactions, the following variable assignment will be used.

Variable	Effect	Indexing
δ_{ϕ_1}	α_h	h=$1,2,3$
δ_{ϕ_2}	β_i	i=$1,2,3$
δ_{θ_1}	ϑ_j	j=$1,2,3$
δ_{θ_2}	γ_k	k=$1,2,3$
Filter Type	λ_ℓ	ℓ=$1,\ldots,8$
Trace Size	τ_m	m=$1,2$

Table 7.4: Greek letter assignment to factor variables.

The interactions will be thought of as a test to compare two differences. In Equation 7.2 the effect of the filter index three, the $Krein\ Kalman_3$ filter, is contrasted against the effect when one level of another factor variable is considered, say the first level of α. We call it the interaction. The overall effect of a factor level will be tested to be equal to the same level, only with another factor variable level expressed simultaneously. This is like looking at the overall effect of a particular filter on MSE, the filter type being the factor level of interest, and comparing it for a particular level of deviation say $\delta_{\phi_1} = -0.5$. For the given example, the comparison would be as follows:

$$(\alpha\lambda)_{13} = (\mu_{......} - \mu_{....3.}) - (\mu_{1.....} - \mu_{1...3.}) \,. \tag{7.2}$$

The only difference in the left difference and right difference is in the expression of the first level of the first factor variable, instead of a sum across the three levels of the first factor variables as in the first term.

This reveals if having a certain type of uncertainty present affects the performance, and if so, how. These comparisons and estimates will only be made for the two best filters, the Bias Corrected EWMA Naive and the Krein space Kalman filter with the exogenous input adjustment and the incorporation of the second uncertainty parameter. These comparisons reveal some information as to why a particular filter is good and how the filter handles different types of uncertainty in parameter estimates. The Classic Kalman filter will be used as a benchmark. In Table 7.5 the interactions of the Kalman filter are presented.

Variable	Lower 95%	Estimate	Upper 95%
$(\alpha\lambda)_{14}$	52.7472	53.1301	53.5129
$(\alpha\lambda)_{24}$	81.8951	82.2779	82.6608
$(\alpha\lambda)_{34}$	87.6063	87.9891	88.3720
$(\beta\lambda)_{14}$	78.2776	78.6605	79.0433
$(\beta\lambda)_{24}$	82.3463	82.7291	83.1119
$(\beta\lambda)_{34}$	61.6248	62.0076	62.3904
$(\vartheta\lambda)_{14}$	77.1614	77.5442	77.9271
$(\vartheta\lambda)_{24}$	77.8081	78.1909	78.5737
$(\vartheta\lambda)_{34}$	67.2792	67.6620	68.0449
$(\gamma\lambda)_{14}$	74.4411	74.8240	75.2068
$(\gamma\lambda)_{24}$	76.8793	77.2621	77.6449
$(\gamma\lambda)_{34}$	70.9283	71.3111	71.6939

Table 7.5: Interactions for *Classic Kalman* and deviation.

The values in Tables 7.6 and 7.7 make apparent that the top two filters here serve to reduce the MSE when in comparison to the Classic Kalman filter. All the values of the classic Kalman filter are of far greater magnitude. One may think it necessary that these two optimal filters have all negative values in these interactions to be truly beneficial. This is not the case with these interactions because the estimates above are two-way interactions that only consider the two factors of filter type and deviation type. The hidden information here is the trace size. In the interpretation

Variable	Lower 95%	Estimate	Upper 95%
$(\alpha\lambda)_{13}$	-21.4633	-21.3356	-21.2080
$(\alpha\lambda)_{23}$	7.6846	7.8122	7.9398
$(\alpha\lambda)_{33}$	13.3958	13.5234	13.6510
$(\beta\lambda)_{13}$	4.0671	4.1947	4.3224
$(\beta\lambda)_{23}$	8.1358	8.2634	8.3910
$(\beta\lambda)_{33}$	-12.5857	-12.4581	-12.3305
$(\vartheta\lambda)_{13}$	2.9509	3.0785	3.2061
$(\vartheta\lambda)_{23}$	3.5975	3.7252	3.8528
$(\vartheta\lambda)_{33}$	-6.9313	-6.8037	-6.6761
$(\gamma\lambda)_{13}$	0.2306	0.3583	0.4859
$(\gamma\lambda)_{23}$	2.6688	2.7964	2.9240
$(\gamma\lambda)_{33}$	-3.2822	-3.1546	-3.0270

Table 7.6: Interactions for $Krein\ Kalman_3$ and deviation.

Variable	Lower 95%	Estimate	Upper 95%
$(\alpha\lambda)_{18}$	-18.1337	-17.7509	-17.3680
$(\alpha\lambda)_{28}$	11.0142	11.3970	11.7798
$(\alpha\lambda)_{38}$	16.7254	17.1082	17.4910
$(\beta\lambda)_{18}$	7.3967	7.7795	8.1624
$(\beta\lambda)_{28}$	11.4653	11.8482	12.2310
$(\beta\lambda)_{38}$	-9.2562	-8.8734	-8.4905
$(\vartheta\lambda)_{18}$	6.2805	6.6633	7.0461
$(\vartheta\lambda)_{28}$	6.9271	7.3099	7.6928
$(\vartheta\lambda)_{38}$	-3.6017	-3.2189	-2.8361
$(\gamma\lambda)_{18}$	3.5602	3.9430	4.3259
$(\gamma\lambda)_{28}$	5.9983	6.3812	6.7640
$(\gamma\lambda)_{38}$	0.0473	0.4301	0.8130

Table 7.7: Interactions for *EWMA B. C. Naive* and deviation.

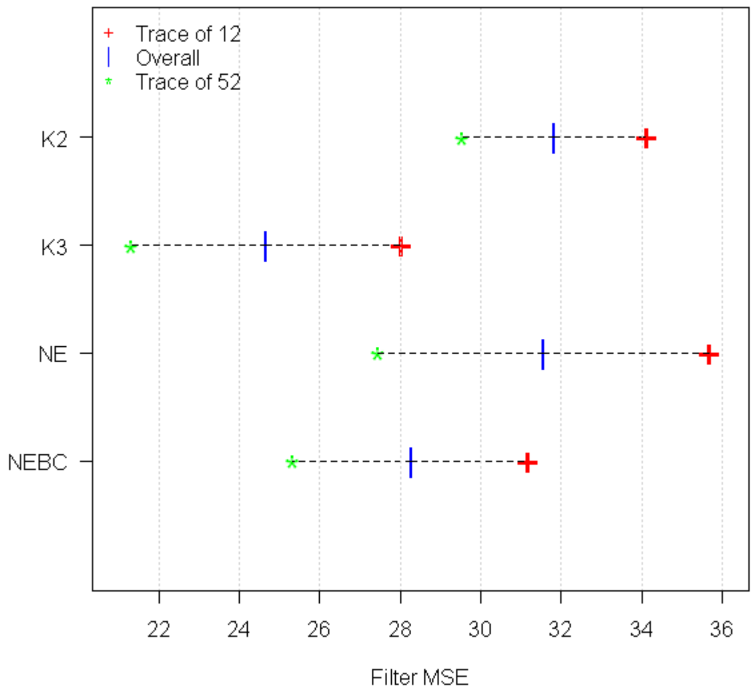


Figure 7.7: Effect of the trace size on filter performance.

of these numbers it is necessary to refer back to the plots and tables of the previous section, specifically Figures 7.4 and 7.6. For the small trace the MSEs are of larger value for the two best filters, (K3 and NEBC), and for the larger trace size the MSEs are smaller. This considered along with the numbers in Tables 7.6 and 7.7 show that the decrease for a trace of size 52 is smaller than the increase for a trace of size 12 in MSE, putting the interaction effect in a slightly positive magnitude for some interactions and a slightly negative magnitude for others. Figure 7.7 is of interest in these regards.

Both filters have a very good reduction in MSE value for a -0.5 deviation in the first AR parameter. However, the Naive type filter has a smaller magnitude effect on MSE. In fact, all of the values of the interactions of the Naive type filter are of smaller magnitude than the Krein Kalman filter. Both filters have large magnitude reductions for the deviations for $\delta_{\phi_1} = -0.5$, $\delta_{\phi_2} = 0.5$, $\delta_{\theta_1} = 0.5$, and $\delta_{\theta_2} = 0.5$. The Krein Kalman filter has an additional small effect for $\delta_{\theta_2} = 0.5$. In comparison to the Classic Kalman filter, both of these filters have much reduced values for all these interactions.

8. CONCLUSIONS

The best two filters out of the eight considered are ranked as follows:

1. Krein space Kalman filter with exogenous input modeling and additional uncertainty adjustment for uncertainty δ_{ϕ_2}

2. Exponentially weighted moving average, bias corrected, Naive filter

Robustness from uncertainty in the model parameter estimates is particularly magnified against the following uncertainties: $\delta_{\phi_1} = -0.5$, $\delta_{\phi_2} = 0.5$, $\delta_{\theta_1} = 0.5$, and $\delta_{\theta_2} = 0.5$. The other uncertainties are far reduced in comparison to the Classic Kalman filter, which is used widely for a state estimate in dynamic feedback control. These top two filters have similar characteristics of robustness to the deviations listed in the previous paragraph, although the Naive has a smaller magnitude robustness with respect to these deviations.

In the previous section it was noted that the Krein space Kalman filter that was presented by Lee et al. (2004) had the widest range of contribution to MSE when comparing a trace of size 12 to size 52. This indicates that although it can be a filter with reasonable robustness level, it suffers most from a lack of consistency. This filter is actually matched by the Classic Kalman filter for a small amount of data. It seems apparent that the adjustments that are expressed in the other two versions of the Krein filter truly improve its performance.

It was also very interesting to see that the Krein Kalman filter with the impulse response modeling capability and the EWMA Naive filter are very similar in performance. When there is a small amount of data, the Kalman type filter outdoes the Naive. When there are at least 52 observations to go on, the tide turns and the EWMA Naive wins out. This version of the Naive filter is very appealing in this presentation because it requires no knowledge whatsoever of the uncertainty. This performance makes this filter stand out among all the filters because of this simplicity. The next best filter that does not account for uncertainty is the Classic Kalman filter, and is not remotely comparable to this filter. An overall conclusion in this respect is that the EWMA Naive filter is truly remarkable and very applicable.

In general, it depends on the ability to model the uncertainty to determine which filter to choose among the top four. These deviations could be estimated using bias. One would need to estimate the bias of the parameters. Future research may lead to an attempt to specify the distribution, and specifically the dispersion of this estimate of bias. If this dispersion leads to a conclusion that deems the knowledge of uncertainty as a shot in the dark so to speak, the EWMA Naive filter may be the best option.

BIBLIOGRAPHY

Akaike, H. (1974), "Markovian Representation of Stochastic Processes, and Its Application to the Analysis of Autoregressive Moving Average Processes," *Annals of the Institute of Statistical Mathematics*, 26, 363–387.

Box, G. and Luceno, A. (1997), *Statistical Control By Monitoring and Feedback Adjustment*, John Wiley & Sons.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994), *Time Series Analysis Forecasting and Control*, Prentice Hall, 3rd ed.

Brockwell, P. J. and Davis, R. A. (2002), *Introduction to Time Series and Forecasting*, Springer, 2nd ed.

Doyle, J. C. (1978), "Guaranteed Margins for LQG Regulators," *IEEE Transactions on Automatic Control*, AC-23, 756–757.

Durbin, J. (1960), "The Fitting of Time-Series Models," *Review of the International Statistical Institute*, 28, 233–244.

Franklin, G. F., Powell, J. D., and Workman, M. L. (1998), *Digital Control of Dynamic Systems*, Addison-Wesley, 3rd ed.

Gomez, V. and Maravall, A. (1994), "Estimation, Prediction, and Interpolation for Nonstationary Series With the Kalman Filter," *Journal of the American Statistical Association*, 89, 611–624.

Hassibi, B., Sayed, A. H., and Kailath, T. (1996a), "Linear Estimation in Krein Spaces-Part I: Theory," *IEEE Transactions on Automatic Control*, 41, 18–33.

— (1996b), "Linear Estimation in Krein Spaces-Part II: Applications," *IEEE Transactions on Automatic Control*, 41, 34–49.

Lee, T. H., Ra, W. S., Yoon, T. S., and Park, J. B. (2004), “Robust Kalman Filtering via Krein Space Estimation,” *IEE Proceedings-Control Theory and Applications*, 151, 59–63.

Maronna, R. A., Martin, R. D., and Yohai, V. J. (2006), *Robust Statistics*, John Wiley & Sons.

Masreliez, C. J. (1975), “Approximate Non-Gaussian Filtering With Linear State and Observation Relations,” *IEEE Transactions on Automatic Control*, AC-20, 107–110.

Pandit, S. M. (1991), *Modal and Spectrum Analysis: Data Dependent Systems in State Space*, John Wiley & Sons.

Pandit, S. M. and Wu, S.-M. (1983), *Time Series and System Analysis with Applications*, John Wiley & Sons.

Petersen, I. R. and Savkin, A. V. (1999), *Robust Kalman Filtering for Signals and Systems with Large Uncertainties*, Bierkhauser.

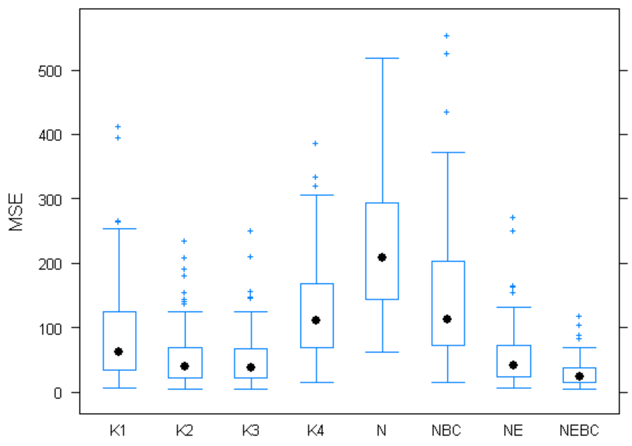
Rencher, A. C. (2000), *Linear Models in Statistics*, John Wiley & Sons.

Shumway, R. H. and Stoffer, D. S. (2006), *Time Series Analysis and Its Applications With R Examples*, Springer, 2nd ed.

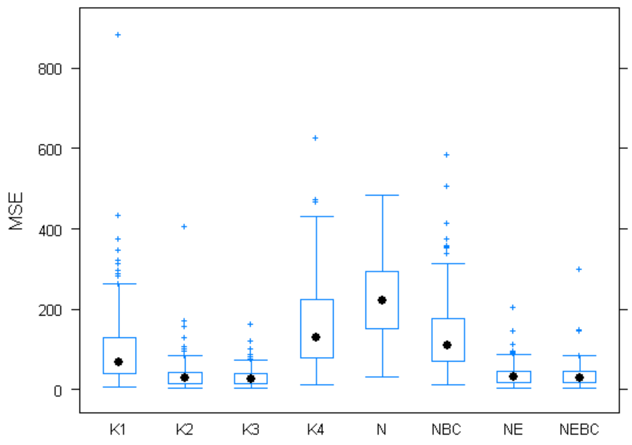
A. MSE BOX PLOTS

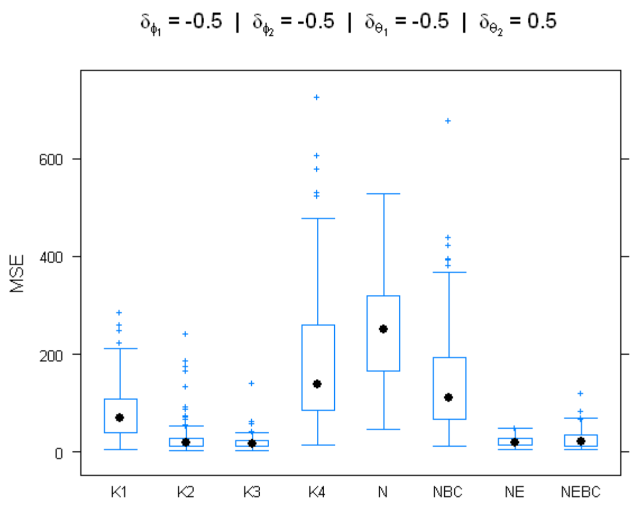
A.1 Trace Size 12

$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$

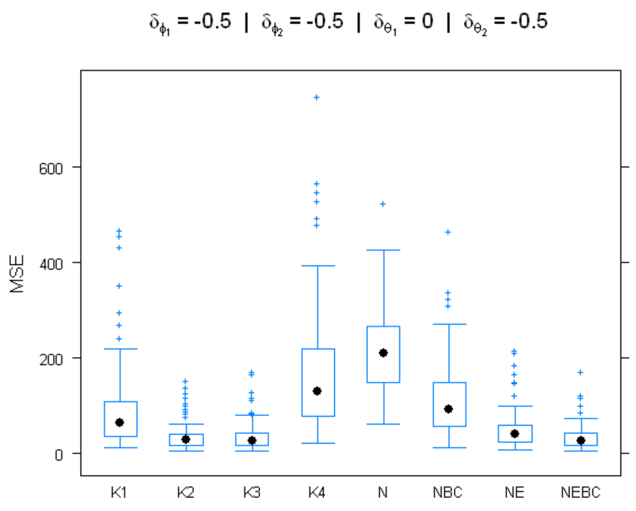


$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$

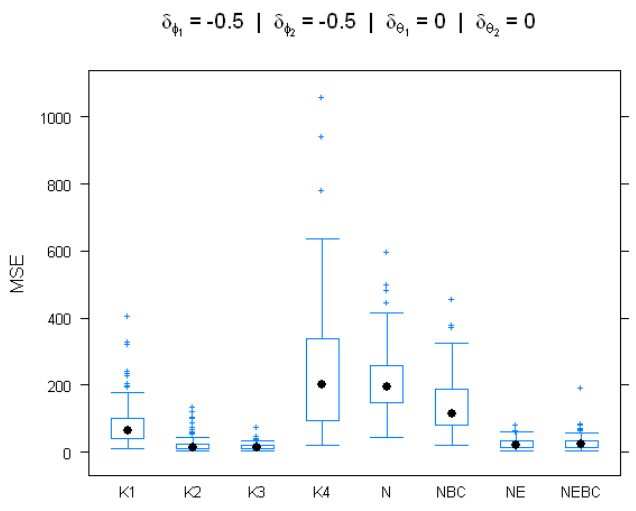




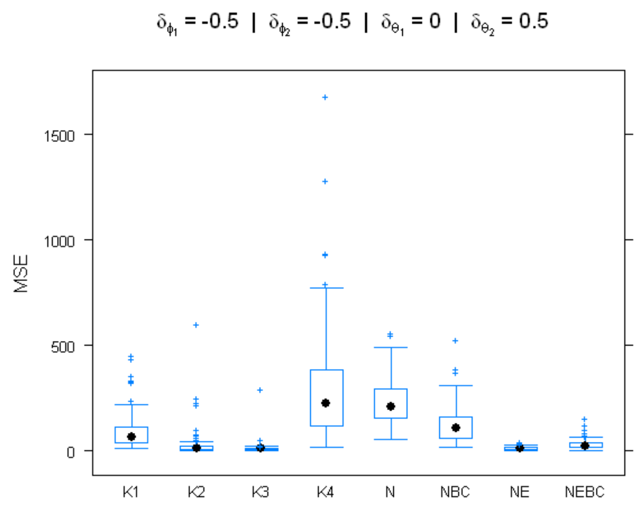
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

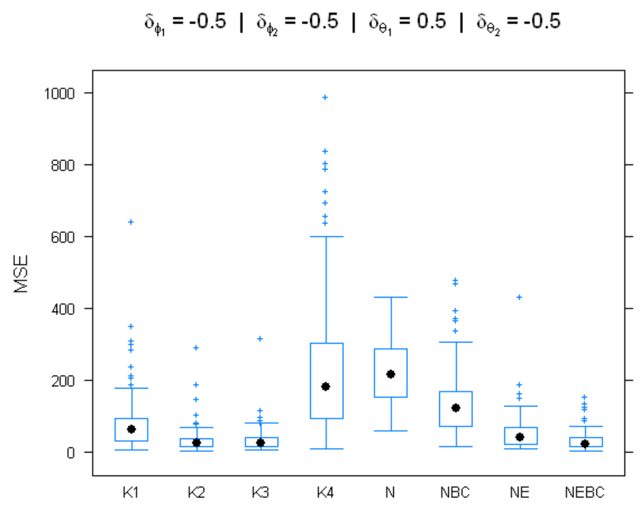


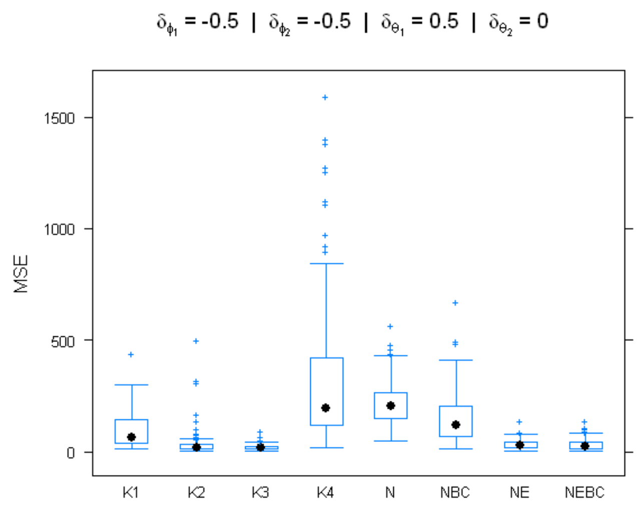
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

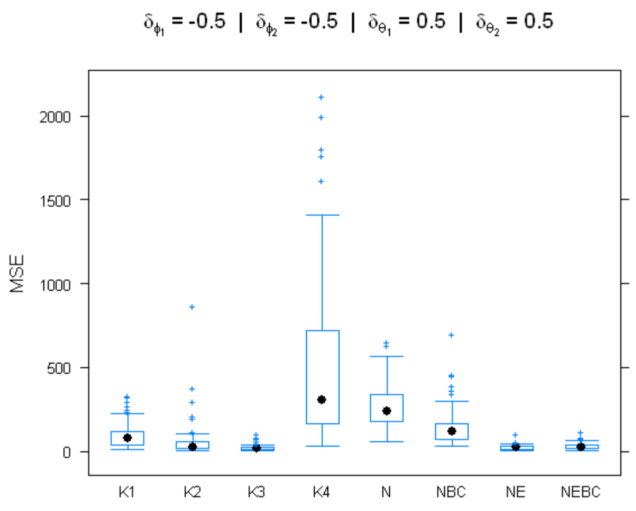


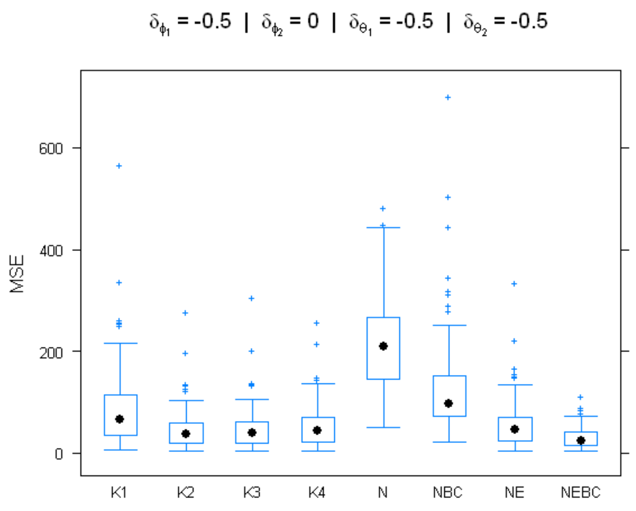
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0$
MSE
0
200
400
600
800
1000
K1
K2
K3
K4
N
NBC
NE
NEBC

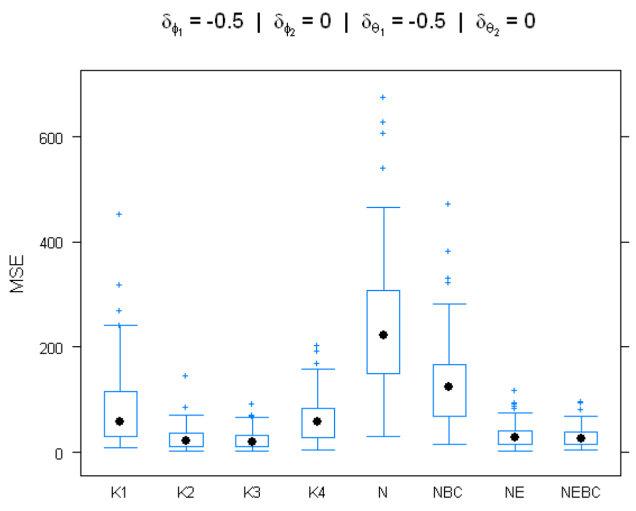

$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$
MSE
0
500
1000
1500
K1
K2
K3
K4
N
NBC
NE
NEBC

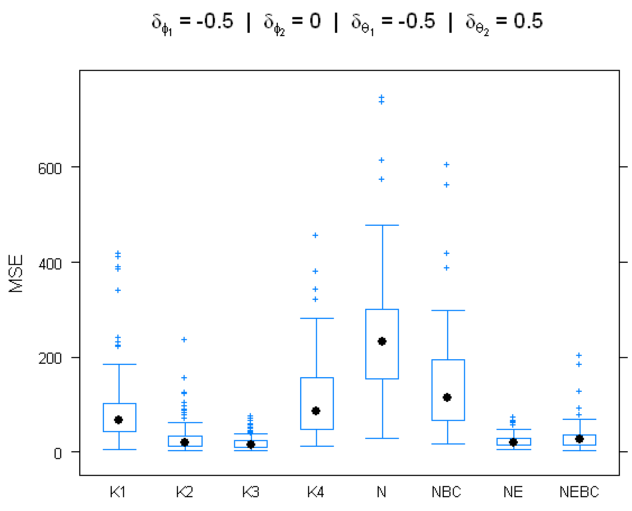

$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
200
400
600
800
1000
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$
MSE
0
500
1000
1500
K1
K2
K3
K4
N
NBC
NE
NEBC

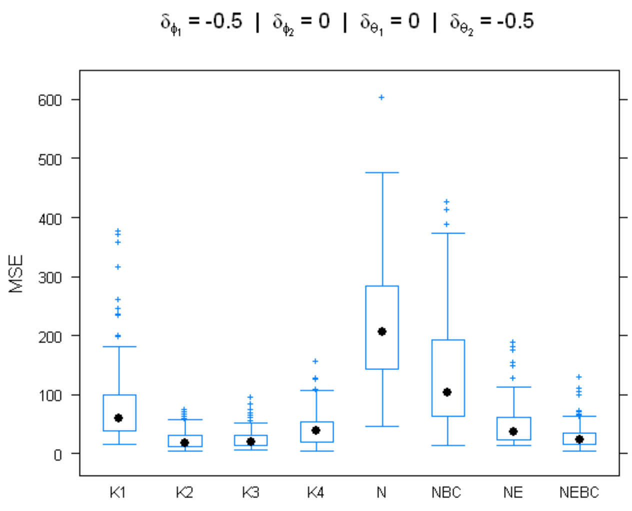

δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = -0.5 | δ_{θ_1} = 0.5 | δ_{θ_2} = 0.5
MSE
0
500
1000
1500
2000
K1
K2
K3
K4
N
NBC
NE
NEBC


δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0 | δ_{θ_1} = -0.5 | δ_{θ_2} = -0.5
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

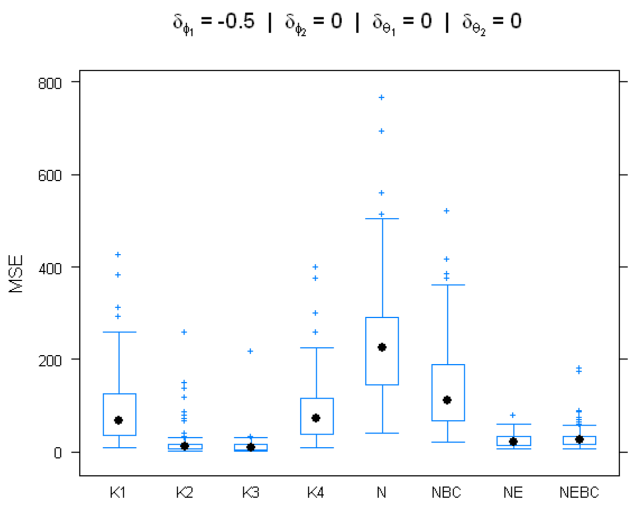

δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0 | δ_{θ_1} = -0.5 | δ_{θ_2} = 0
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC



δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0 | δ_{θ_1} = -0.5 | δ_{θ_2} = 0.5
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

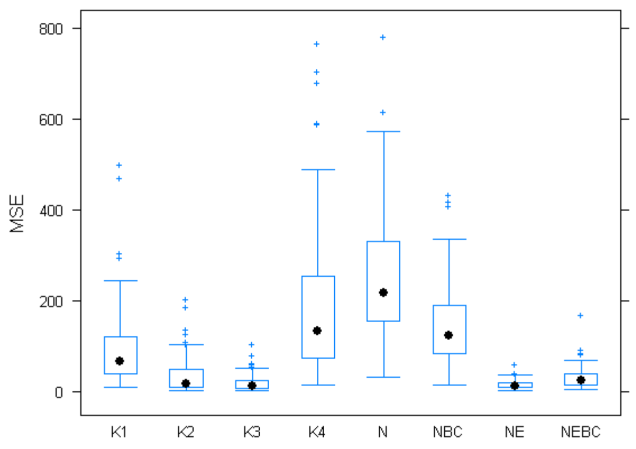


δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0 | δ_{θ_1} = 0 | δ_{θ_2} = -0.5
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC

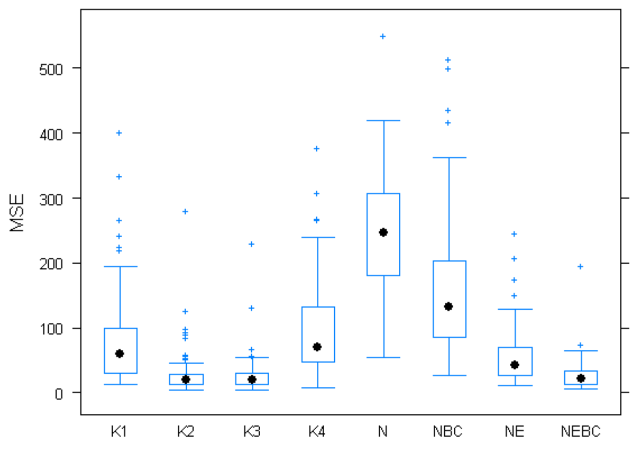


δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0 | δ_{θ_1} = 0 | δ_{θ_2} = 0
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC

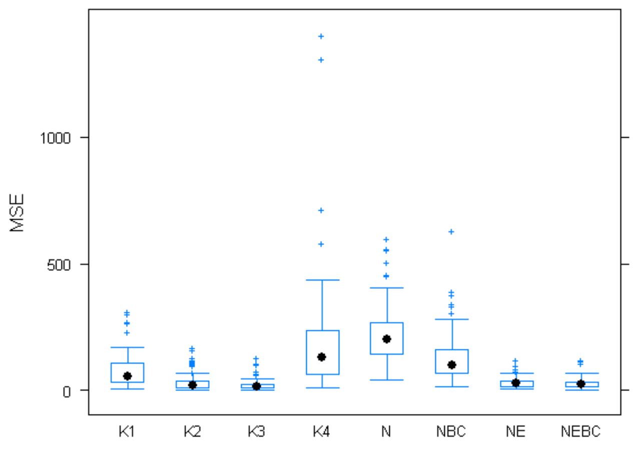
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$

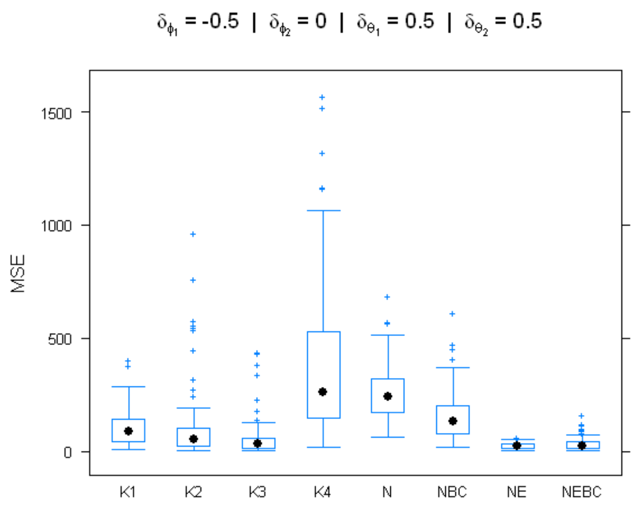


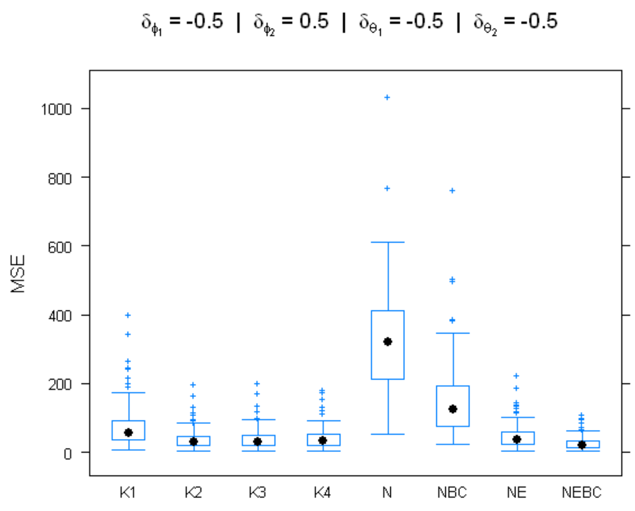
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$

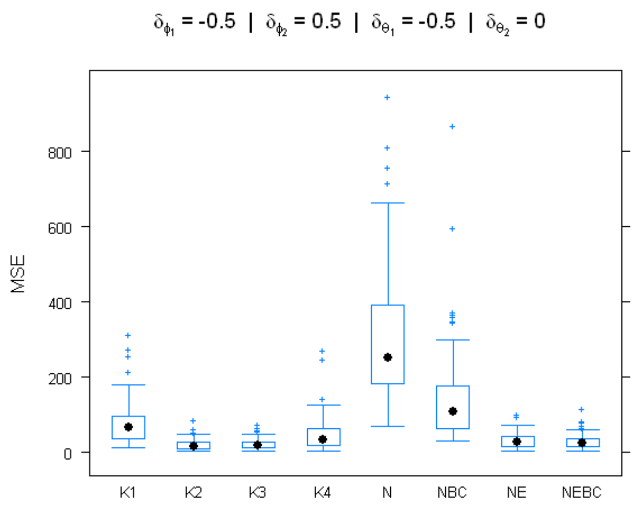


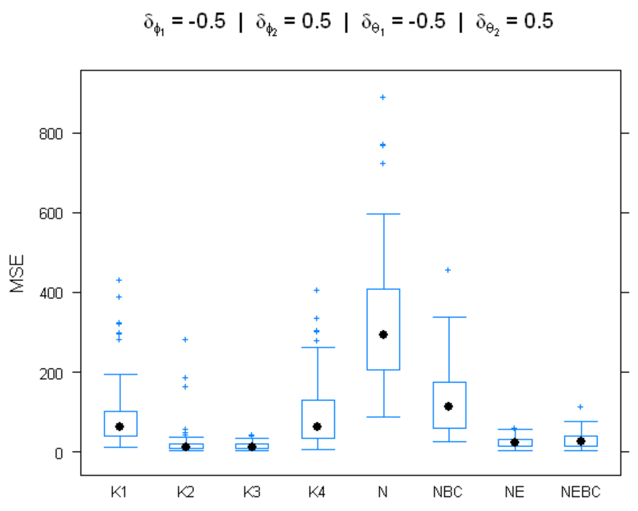
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$

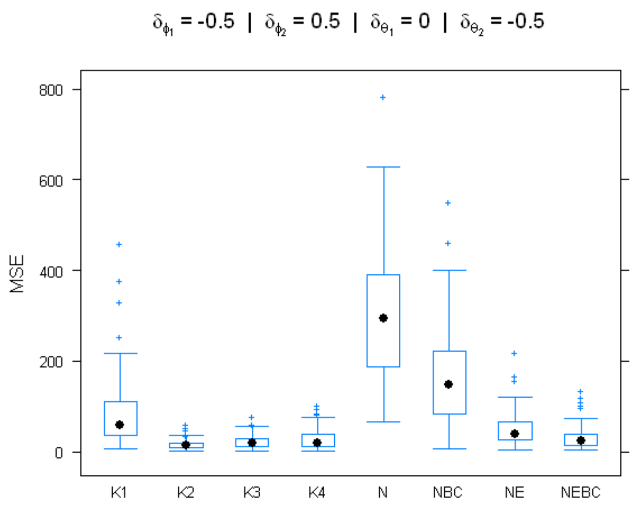


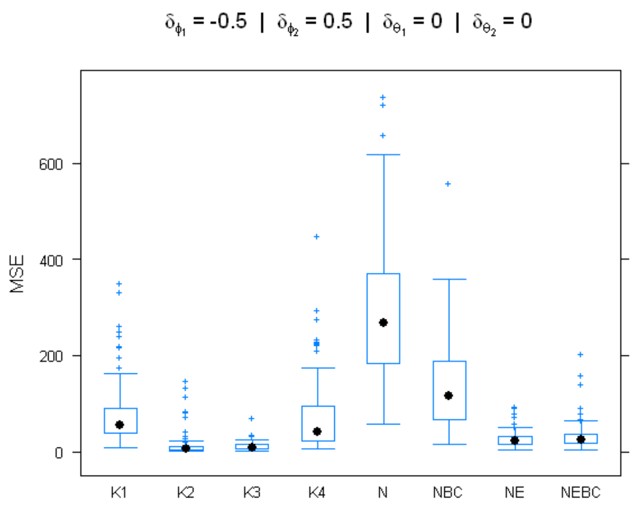

δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0 | δ_{θ_1} = 0.5 | δ_{θ_2} = 0.5
MSE
1500
1000
500
0
K1
K2
K3
K4
N
NBC
NE
NEBC

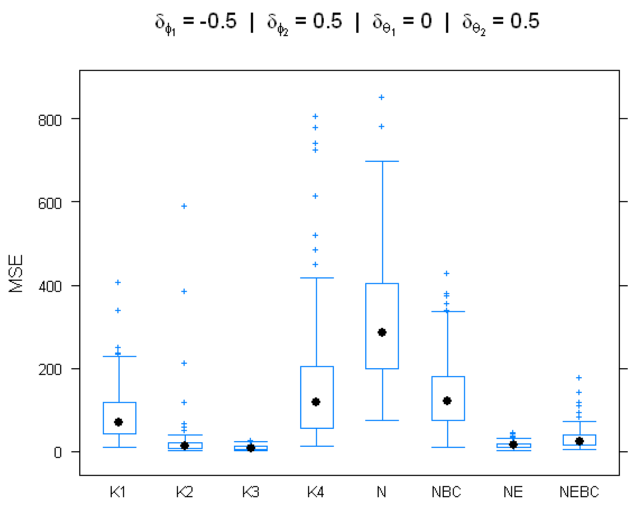

δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = -0.5 | δ_{θ_2} = -0.5
MSE
1000
800
600
400
200
0
K1
K2
K3
K4
N
NBC
NE
NEBC

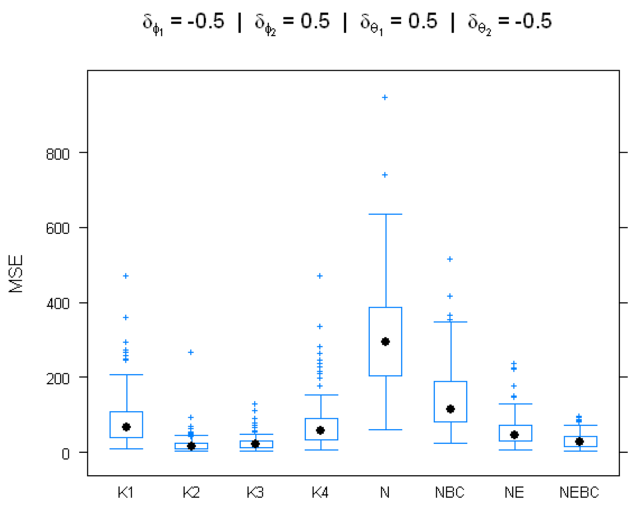

δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = -0.5 | δ_{θ_2} = 0
MSE
800
600
400
200
0
K1
K2
K3
K4
N
NBC
NE
NEBC

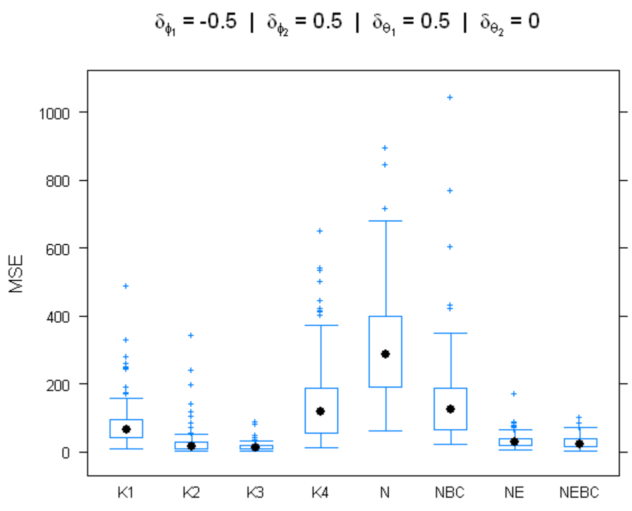

δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = -0.5 | δ_{θ_2} = 0.5
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC


δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = 0 | δ_{θ_2} = -0.5
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC

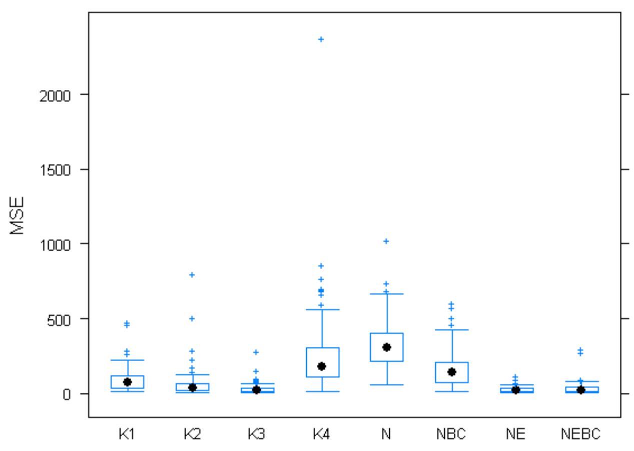

δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = 0 | δ_{θ_2} = 0
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC


δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = 0 | δ_{θ_2} = 0.5
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC

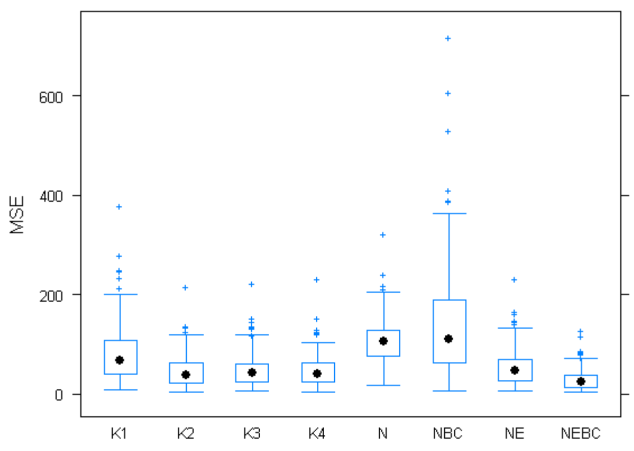

δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = 0.5 | δ_{θ_2} = -0.5
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC


δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = 0.5 | δ_{θ_2} = 0
MSE
0
200
400
600
800
1000
K1
K2
K3
K4
N
NBC
NE
NEBC

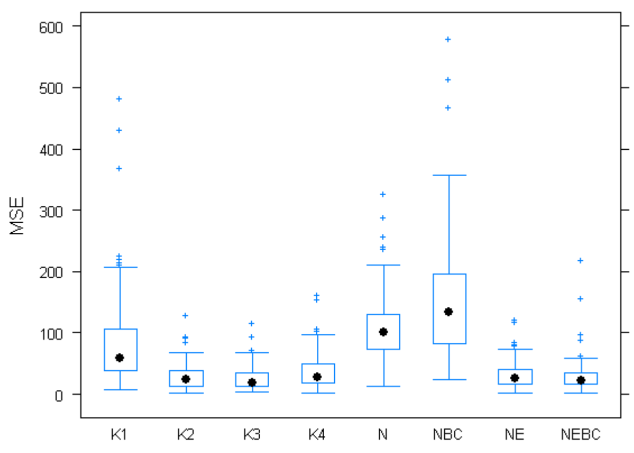
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$



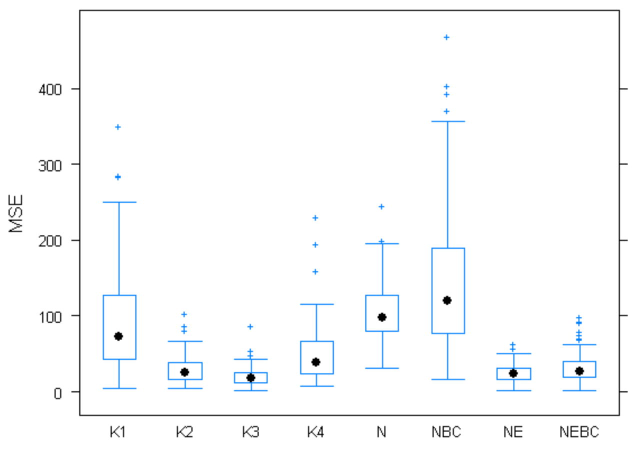
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$



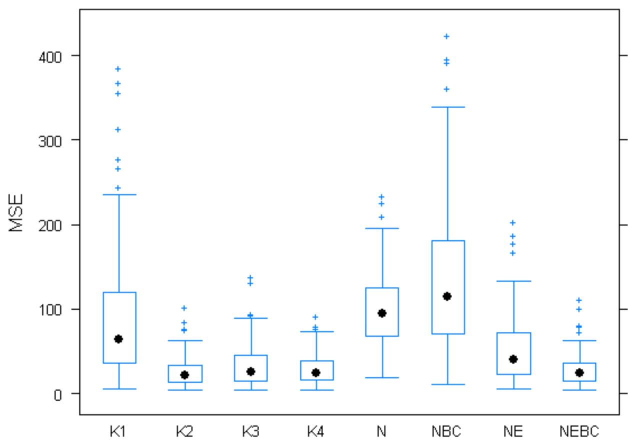
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$



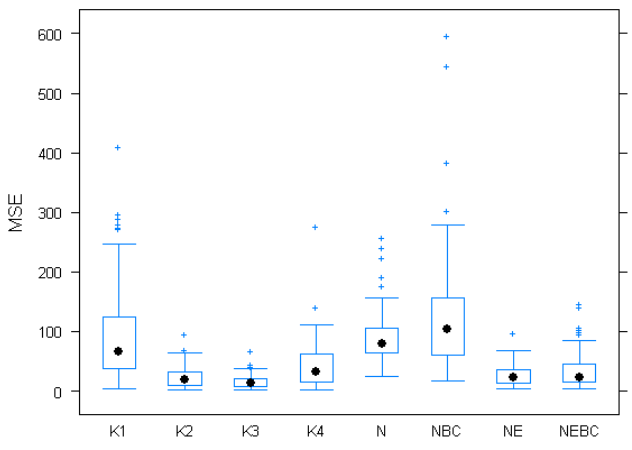
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\Theta_1} = -0.5$ | $\delta_{\Theta_2} = 0.5$

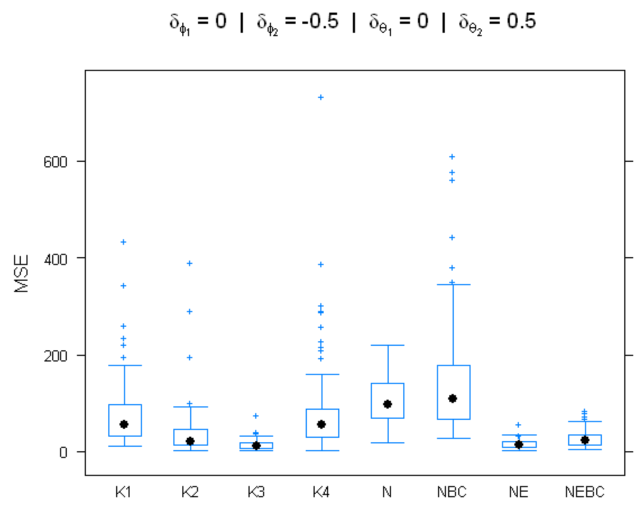


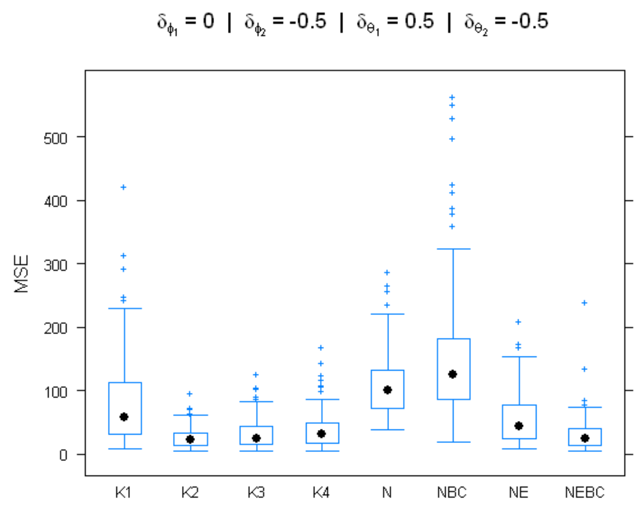
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\Theta_1} = 0$ | $\delta_{\Theta_2} = -0.5$

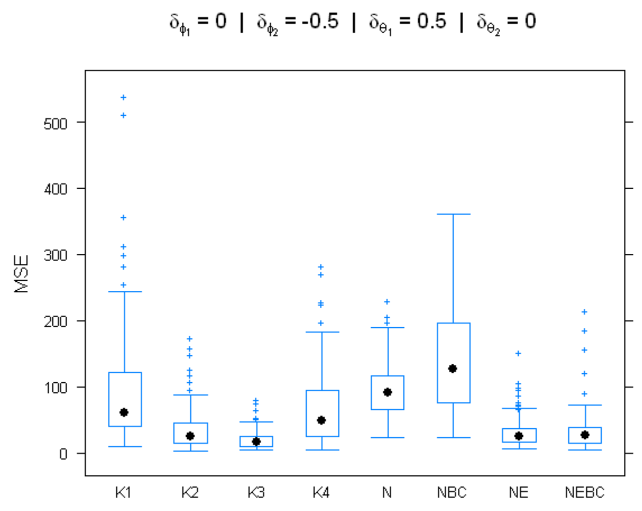


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\Theta_1} = 0$ | $\delta_{\Theta_2} = 0$

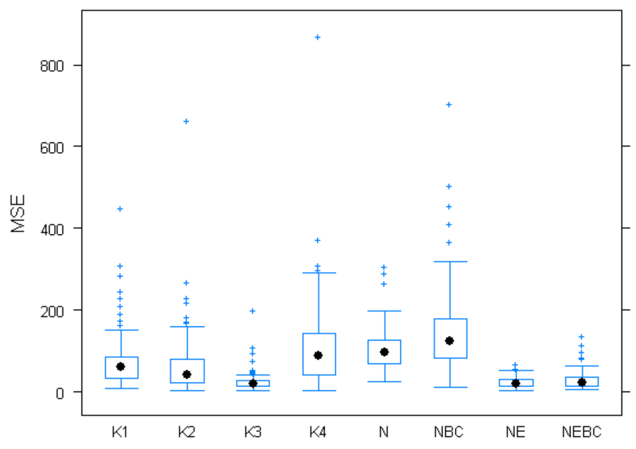



$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

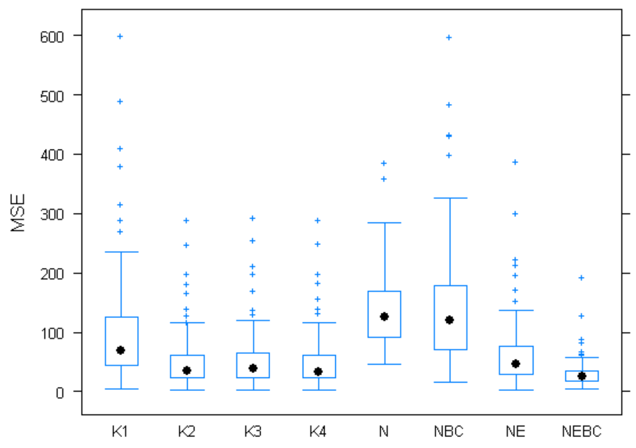

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
500
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
400
500
K1
K2
K3
K4
N
NBC
NE
NEBC

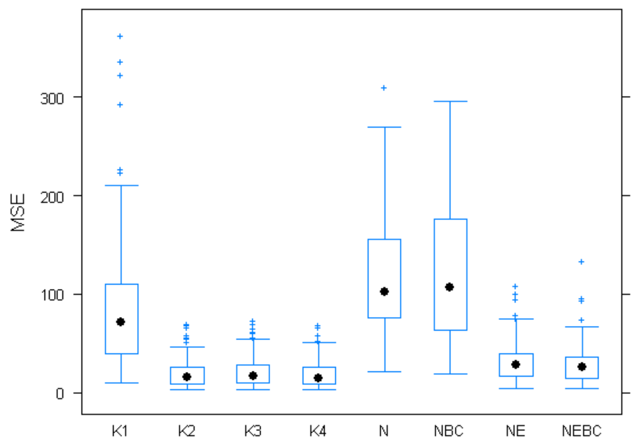
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$

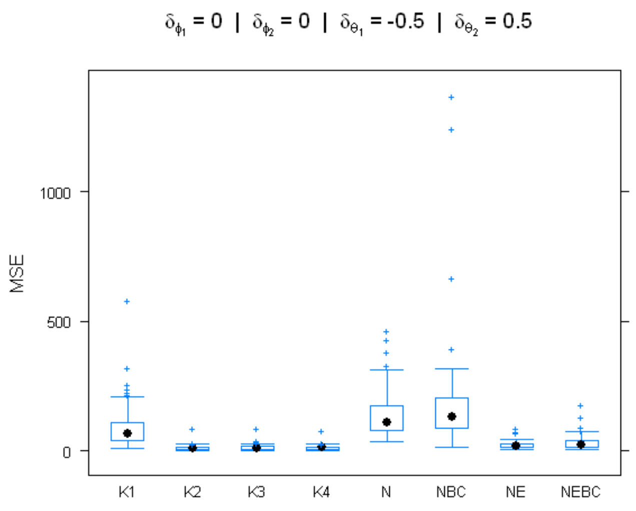


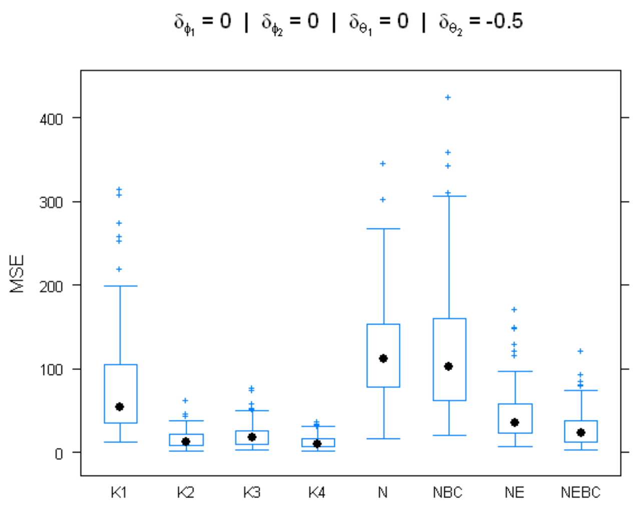
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$

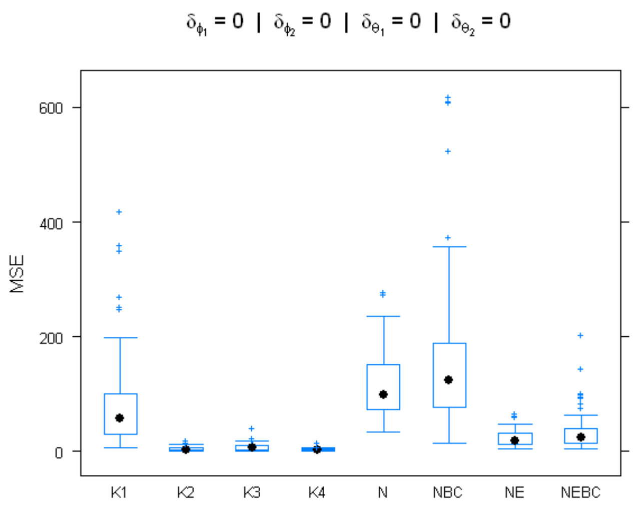


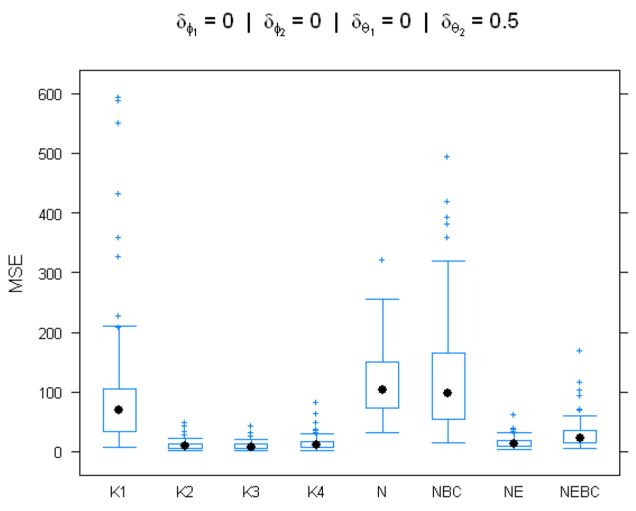
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$



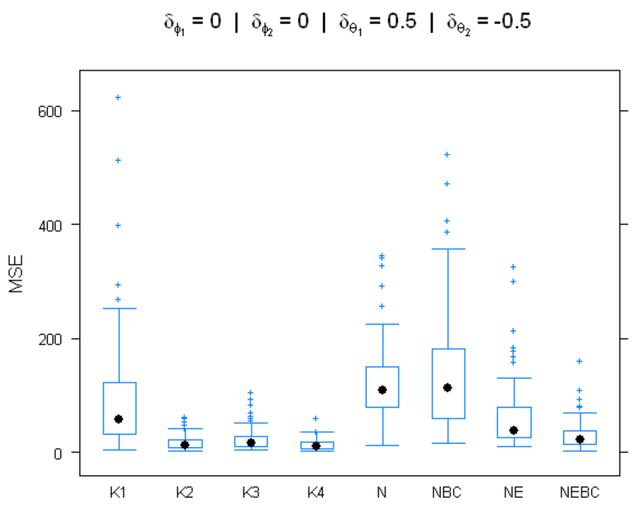

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$
MSE
1000
500
0
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$
MSE
400
300
200
100
0
K1
K2
K3
K4
N
NBC
NE
NEBC

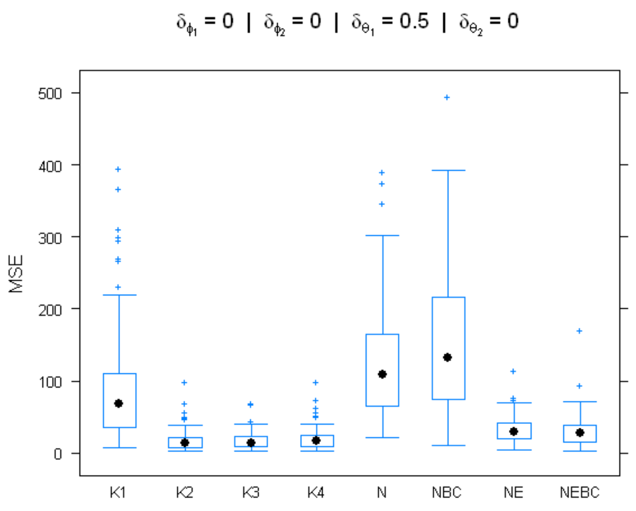

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0$
MSE
600
400
200
0
K1
K2
K3
K4
N
NBC
NE
NEBC



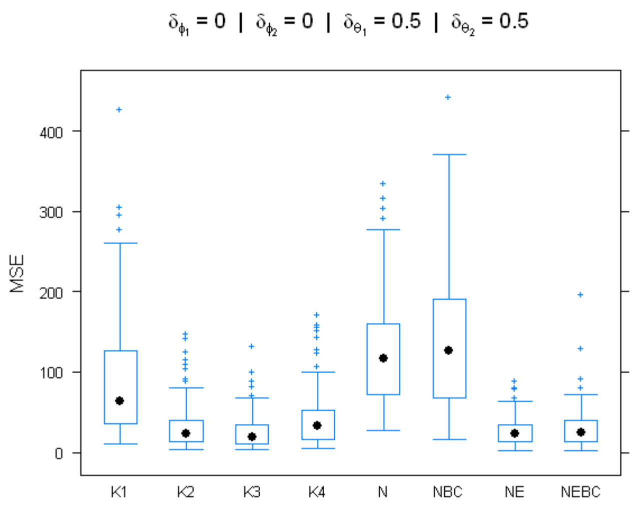
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC



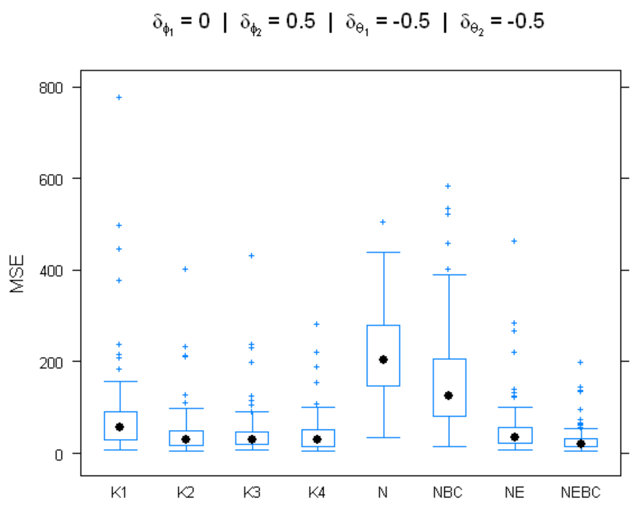
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC



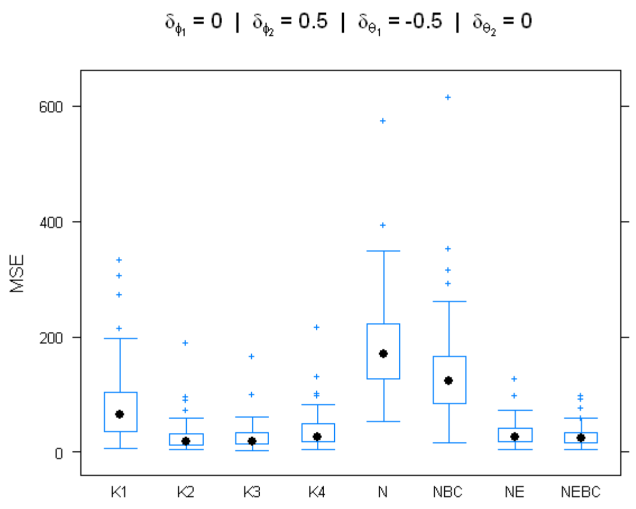
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
400
500
K1
K2
K3
K4
N
NBC
NE
NEBC



$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

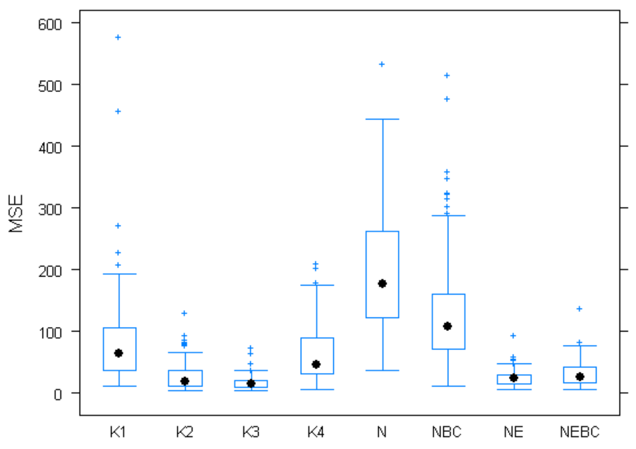


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC

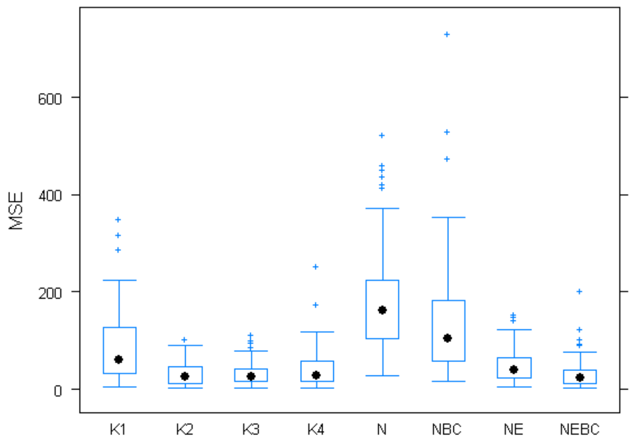


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

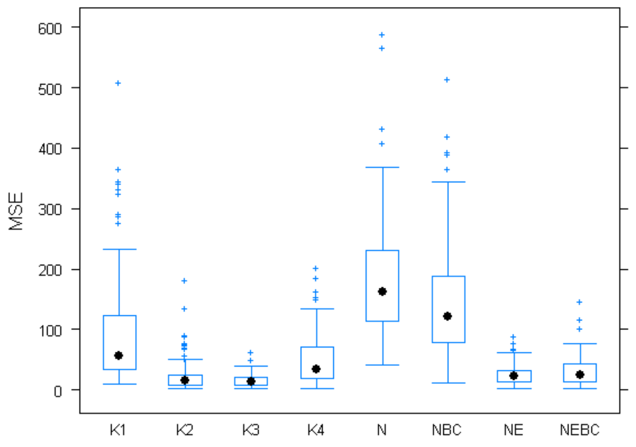
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$

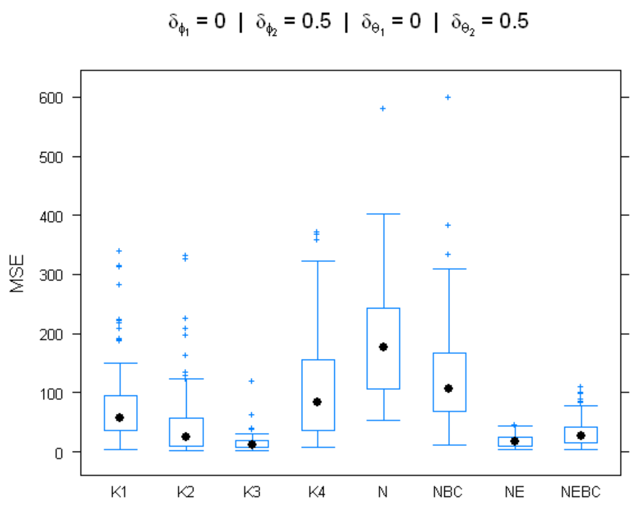


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$

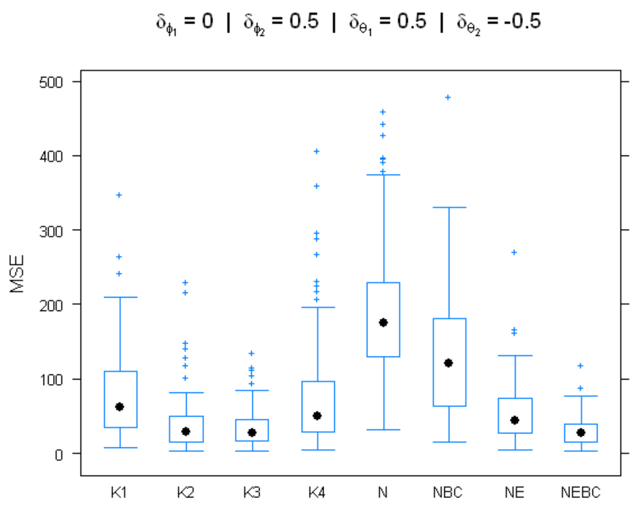


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0$

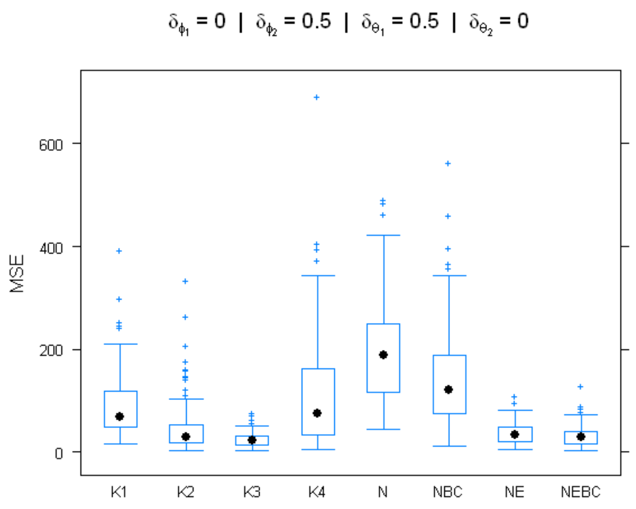




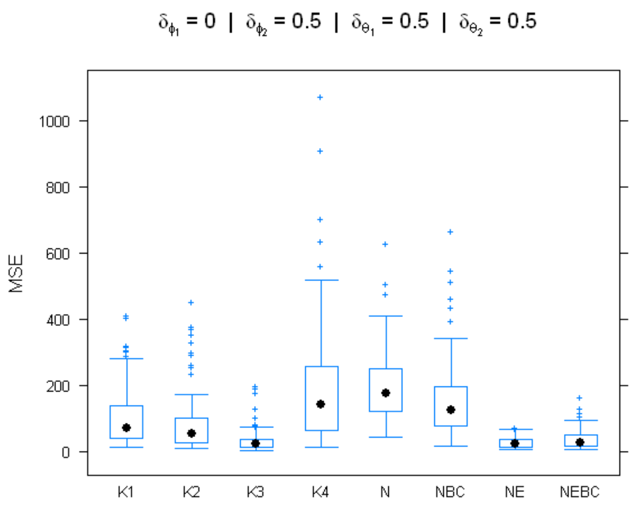
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC

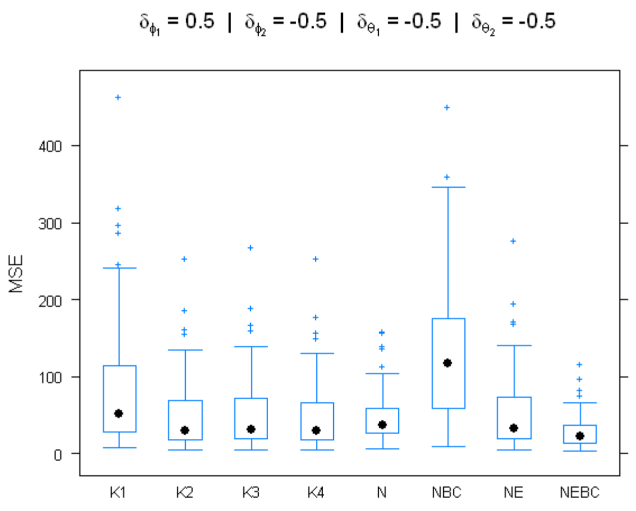


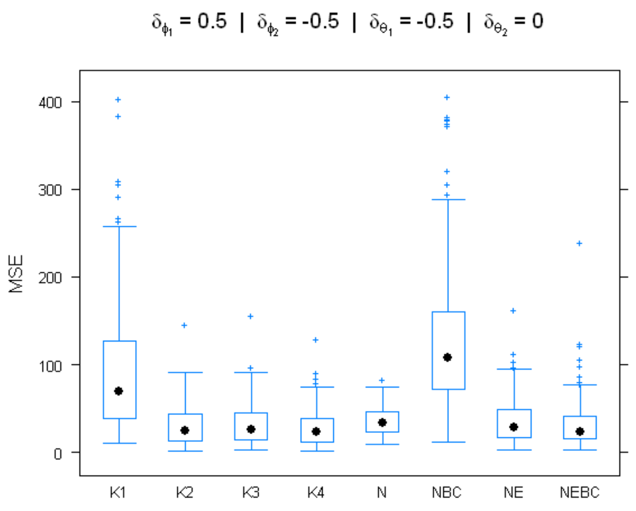
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
500
K1
K2
K3
K4
N
NBC
NE
NEBC

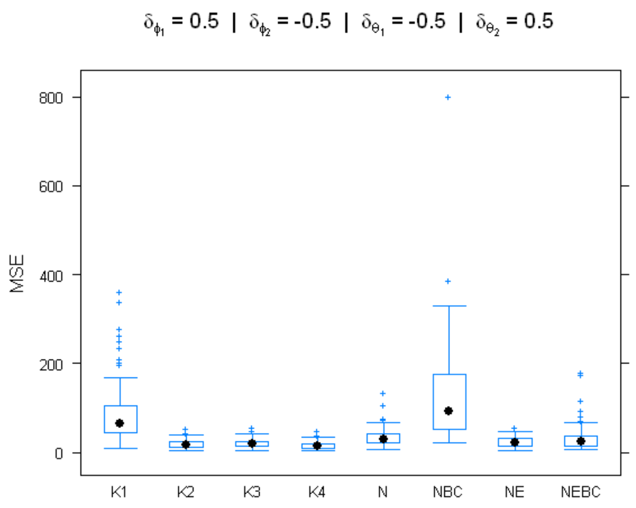


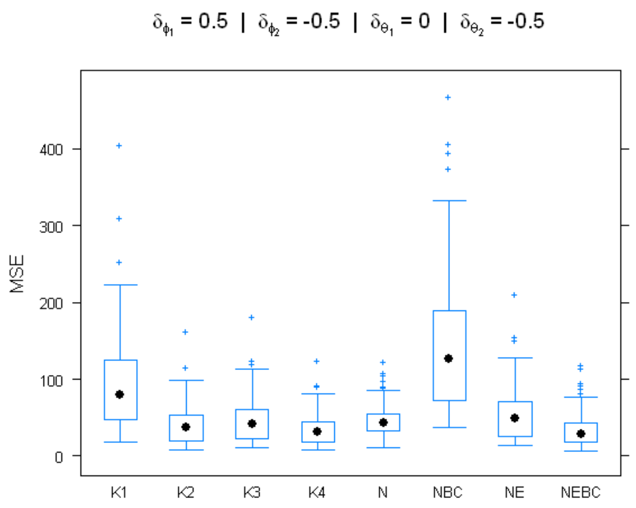
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

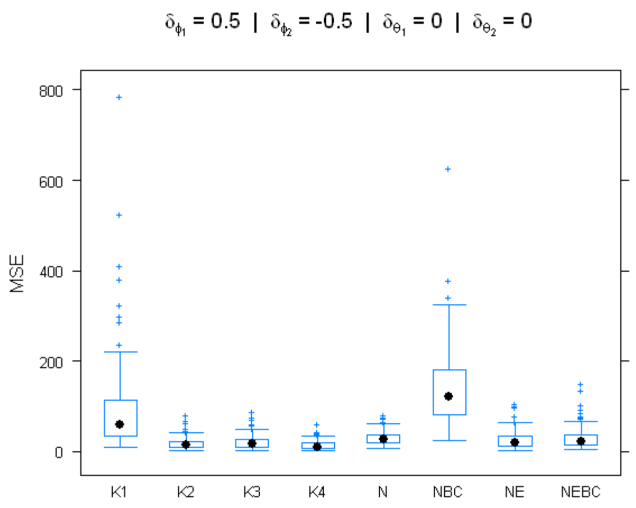

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
200
400
600
800
1000
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

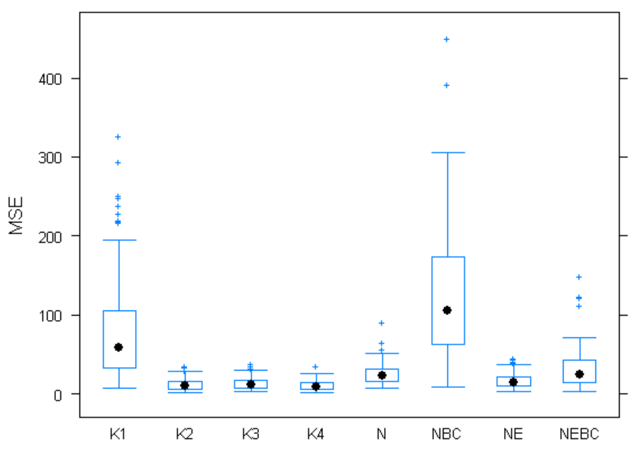

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC


δ_φ1 = 0.5 | δ_φ2 = -0.5 | δ_θ1 = -0.5 | δ_θ2 = 0.5
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC

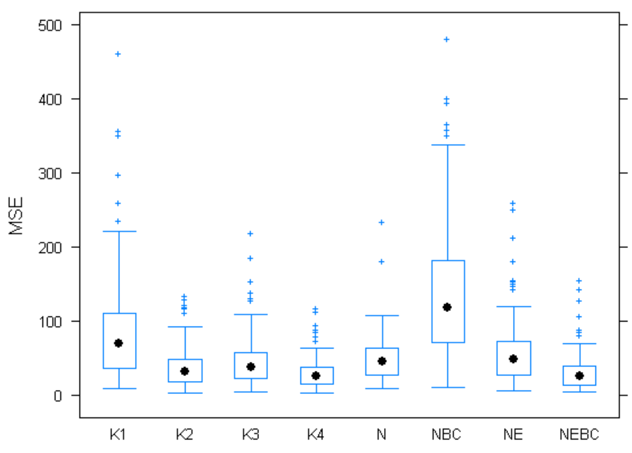

δ_φ1 = 0.5 | δ_φ2 = -0.5 | δ_θ1 = 0 | δ_θ2 = -0.5
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC


δ_φ1 = 0.5 | δ_φ2 = -0.5 | δ_θ1 = 0 | δ_θ2 = 0
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC

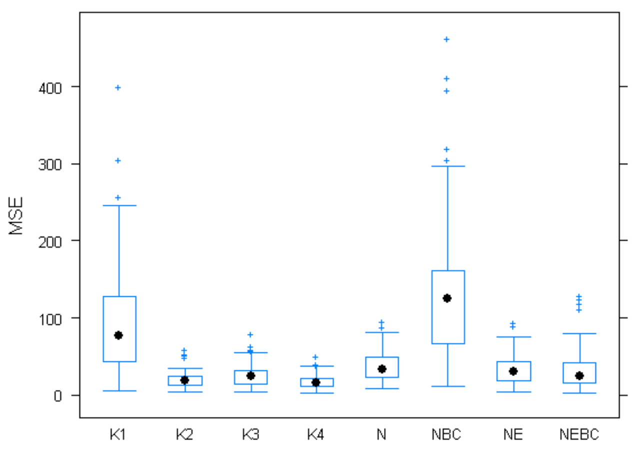
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$



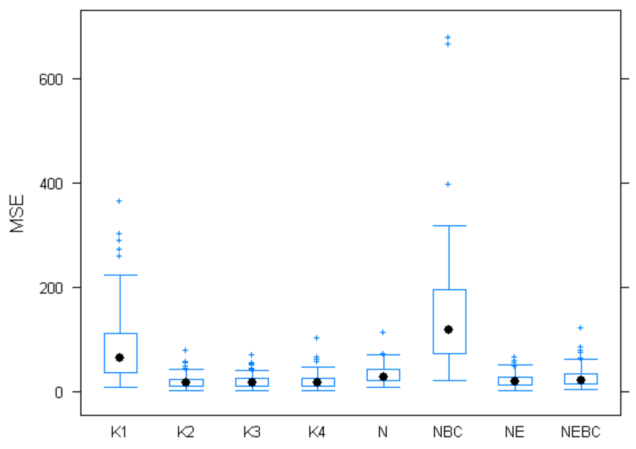
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$



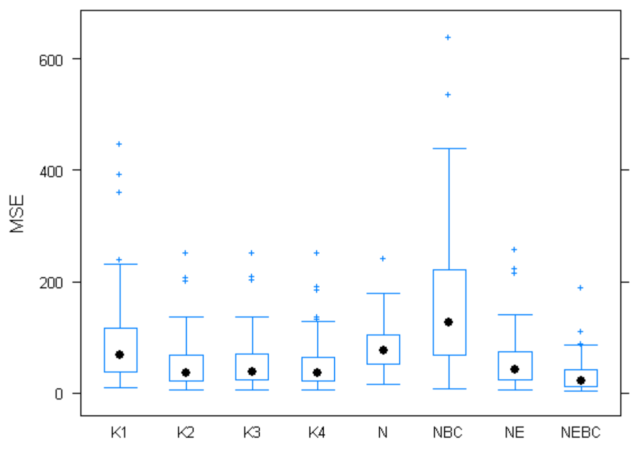
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$



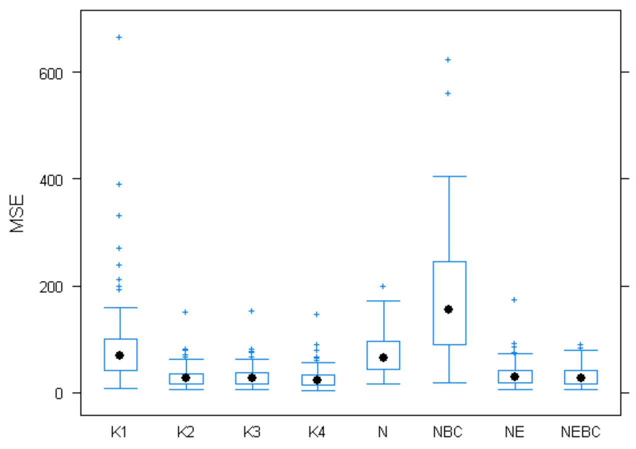
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$



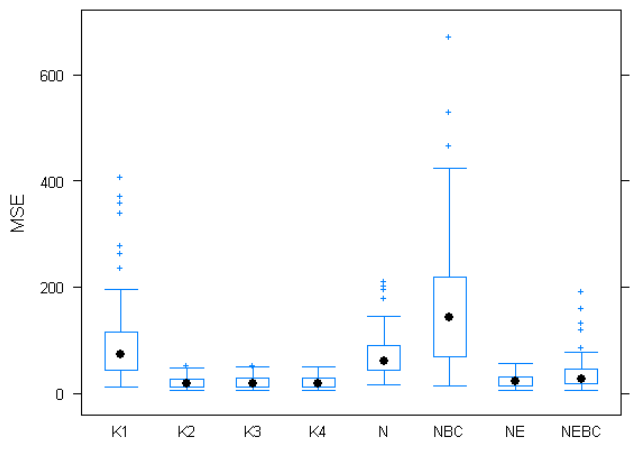
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$



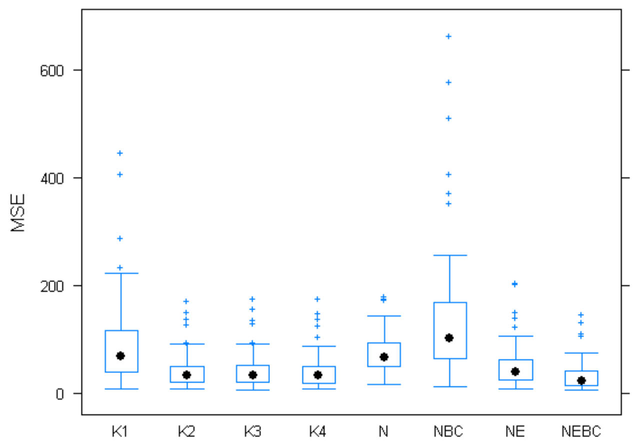
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$



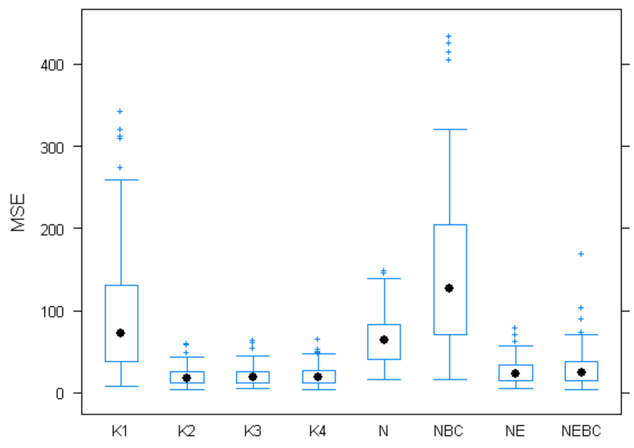
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$



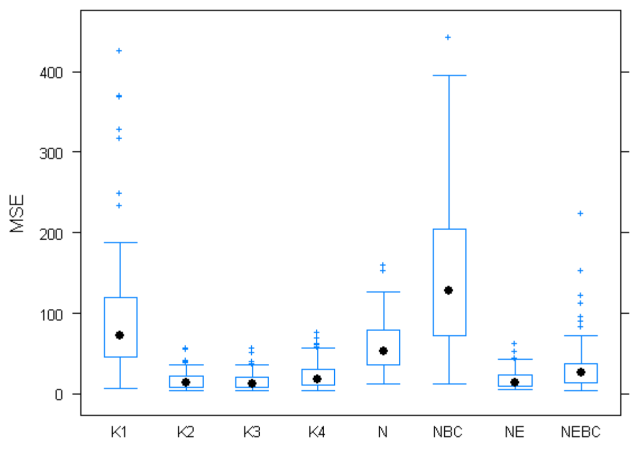
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$



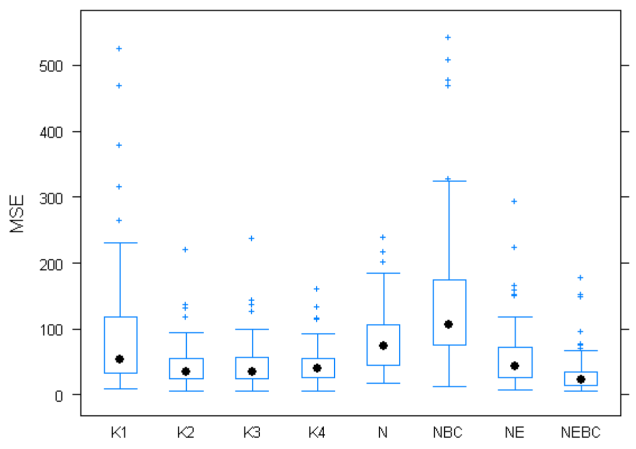
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0$



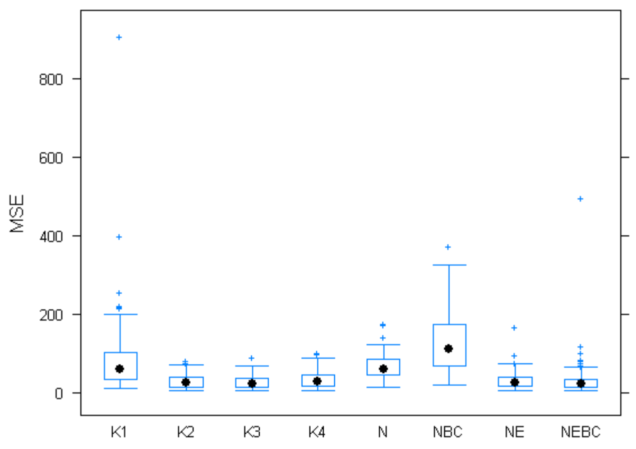
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$



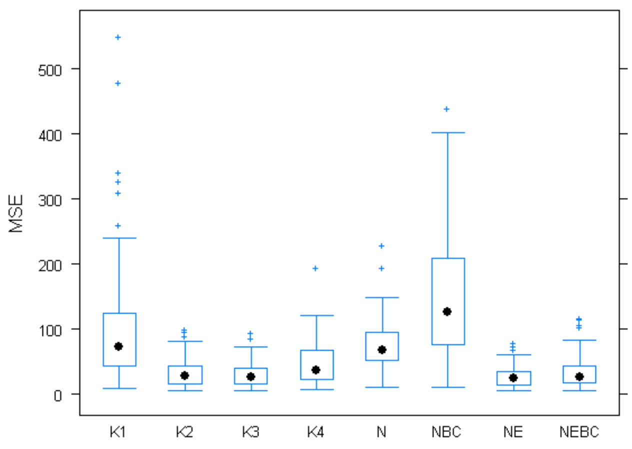
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$



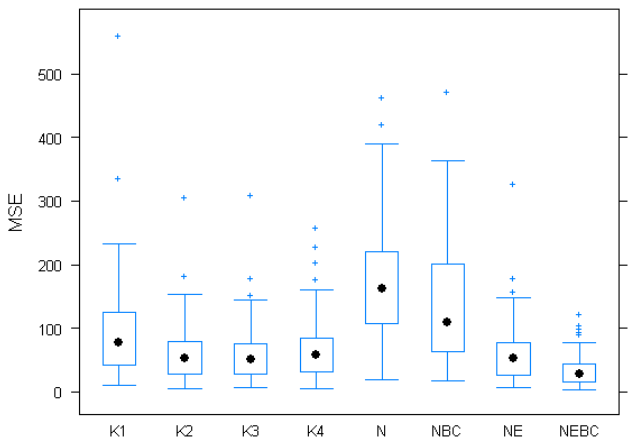
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$



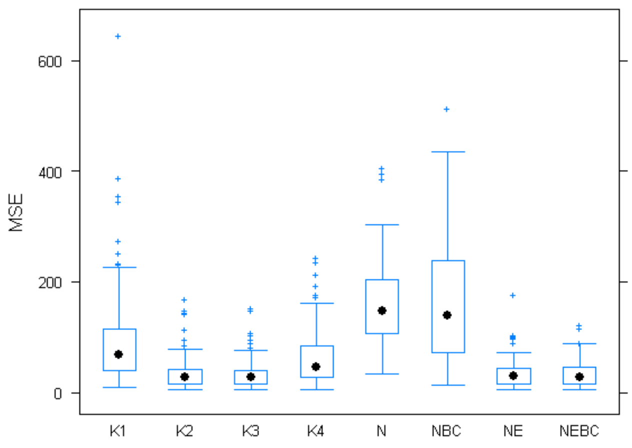
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\Theta_1} = 0.5$ | $\delta_{\Theta_2} = 0.5$



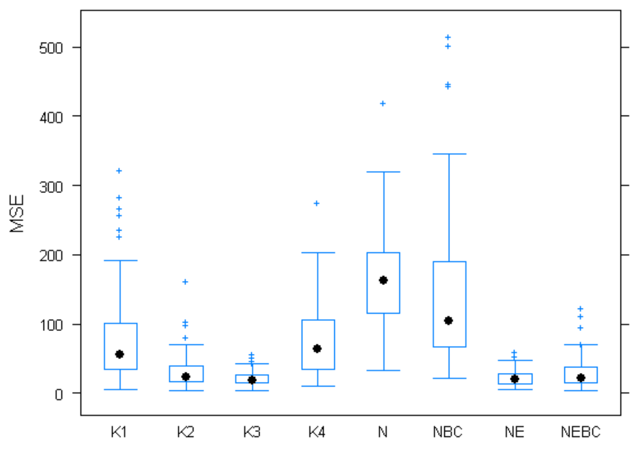
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = -0.5$ | $\delta_{\Theta_2} = -0.5$



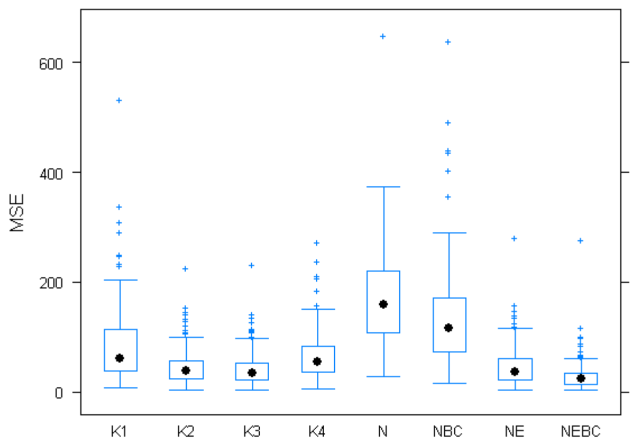
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = -0.5$ | $\delta_{\Theta_2} = 0$



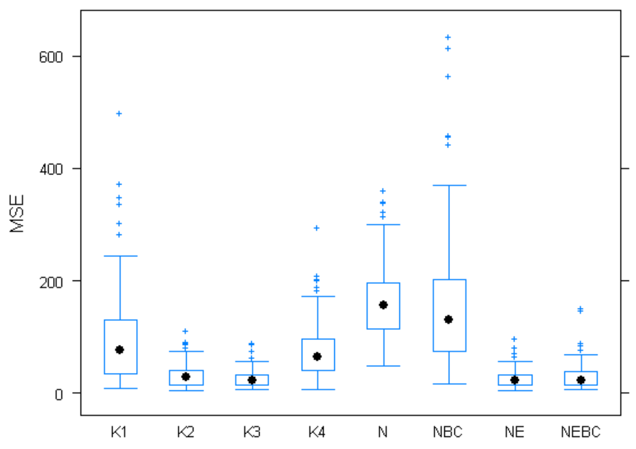
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = -0.5$ | $\delta_{\Theta_2} = 0.5$

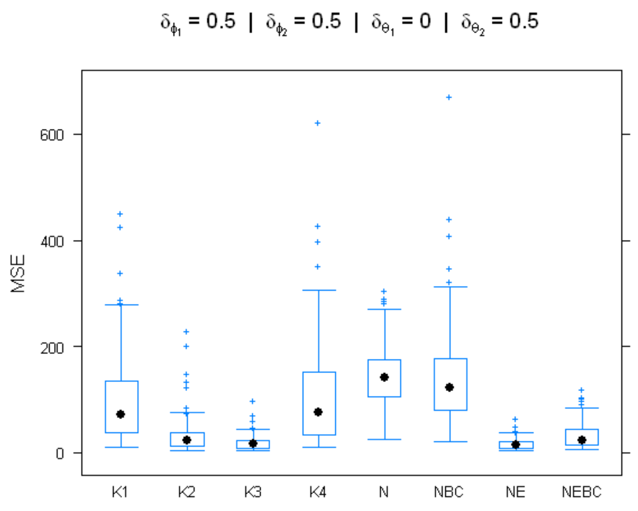


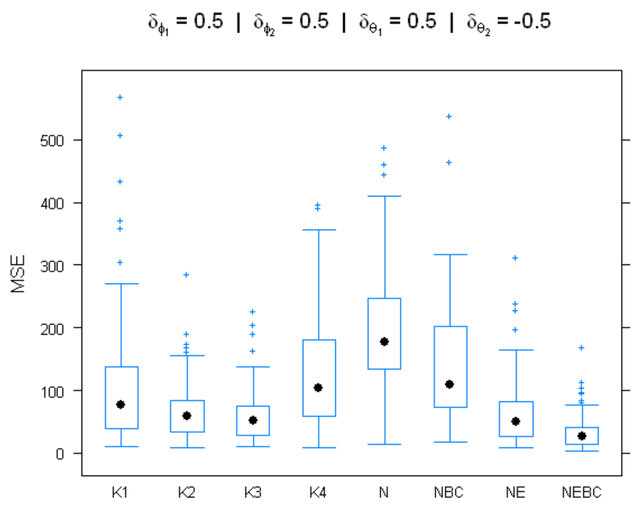
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = 0$ | $\delta_{\Theta_2} = -0.5$

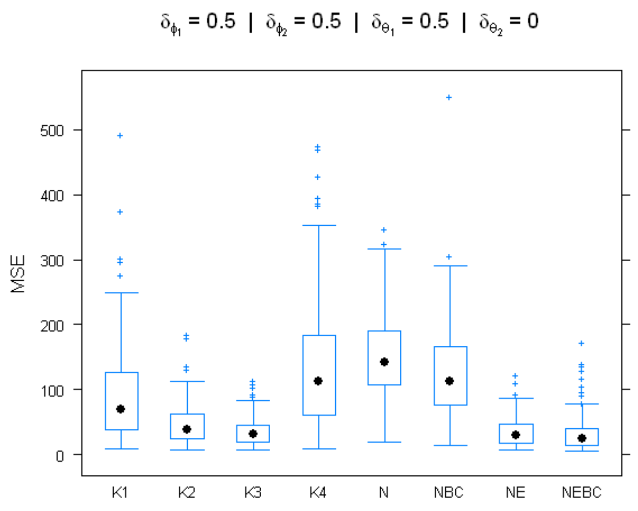


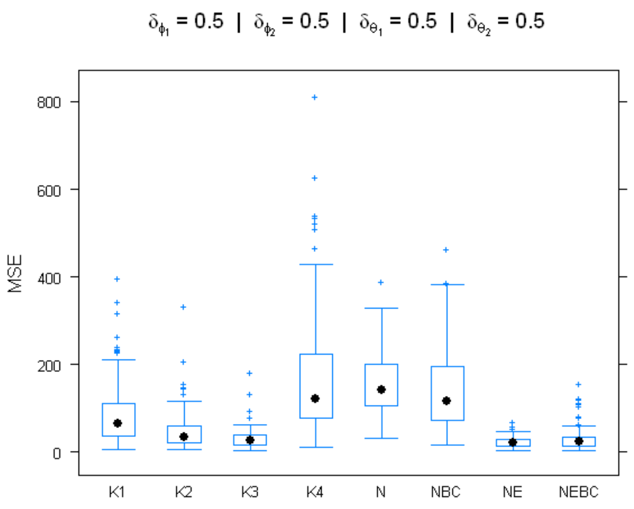
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = 0$ | $\delta_{\Theta_2} = 0$



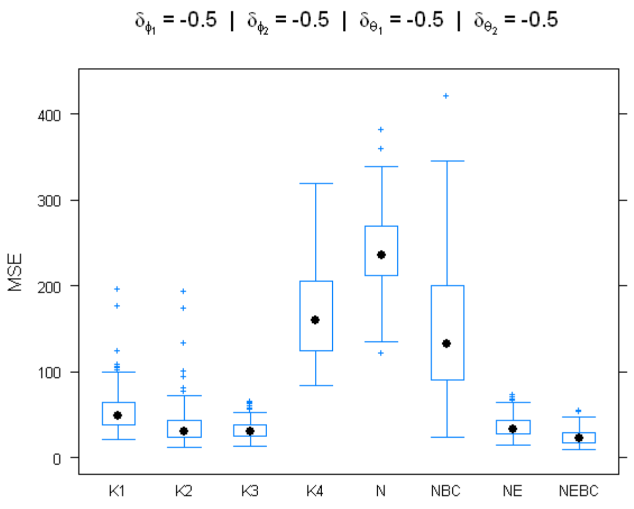

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = 0$ | $\delta_{\Theta_2} = 0.5$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

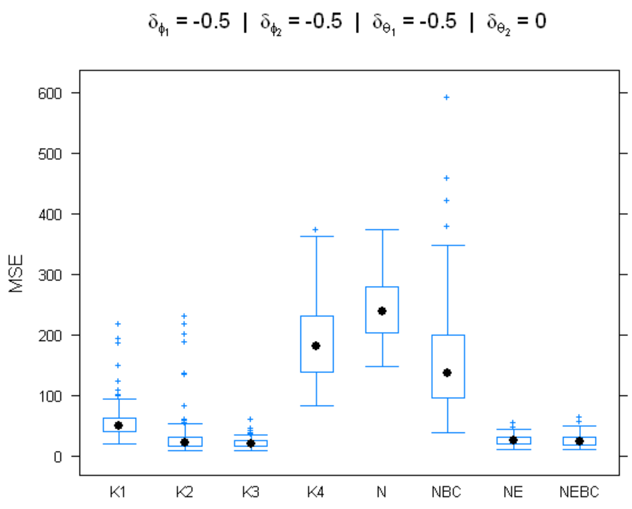

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = 0.5$ | $\delta_{\Theta_2} = -0.5$
MSE
0
100
200
300
400
500
K1
K2
K3
K4
N
NBC
NE
NEBC

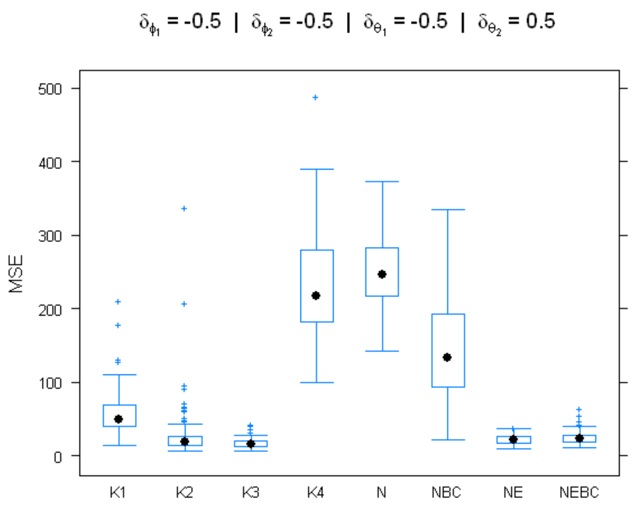

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = 0.5$ | $\delta_{\Theta_2} = 0$
MSE
0
100
200
300
400
500
K1
K2
K3
K4
N
NBC
NE
NEBC

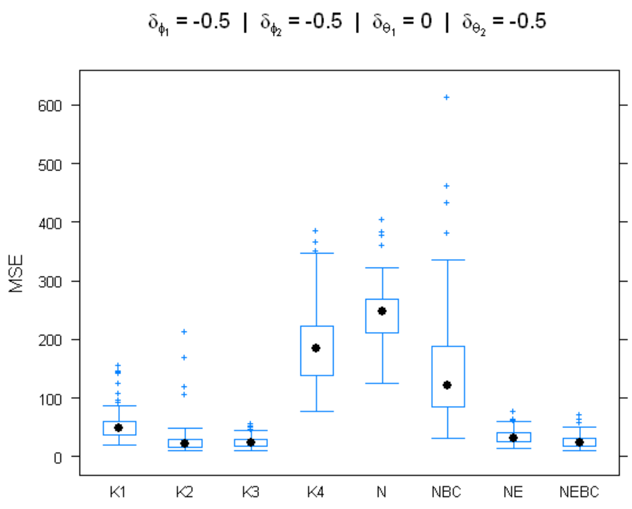


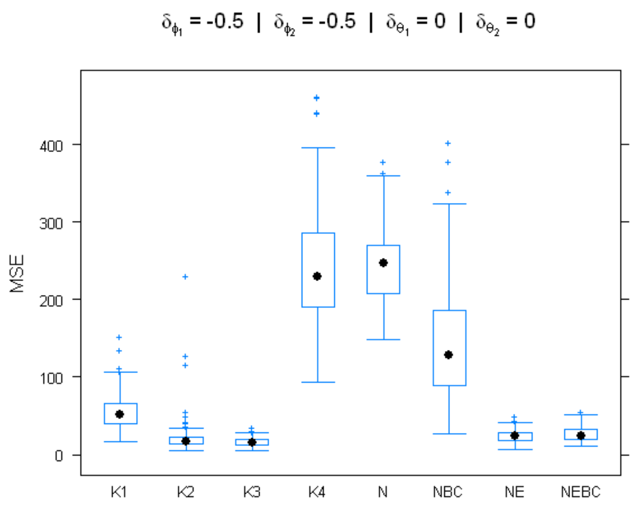
A.2 Trace Size 52



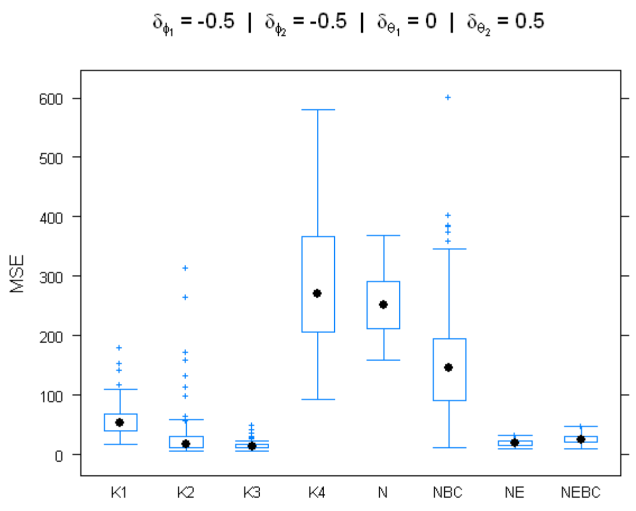

$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
100
200
300
400
500
K1
K2
K3
K4
N
NBC
NE
NEBC

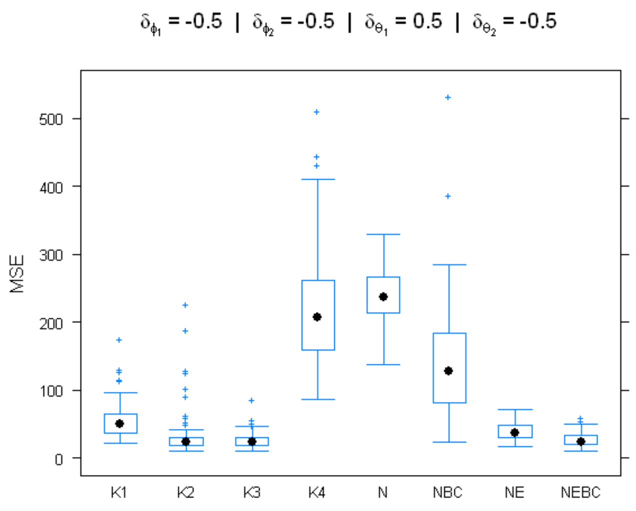

$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC



δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = -0.5 | δ_{θ_1} = 0 | δ_{θ_2} = 0
MSE
400
300
200
100
0
K1
K2
K3
K4
N
NBC
NE
NEBC

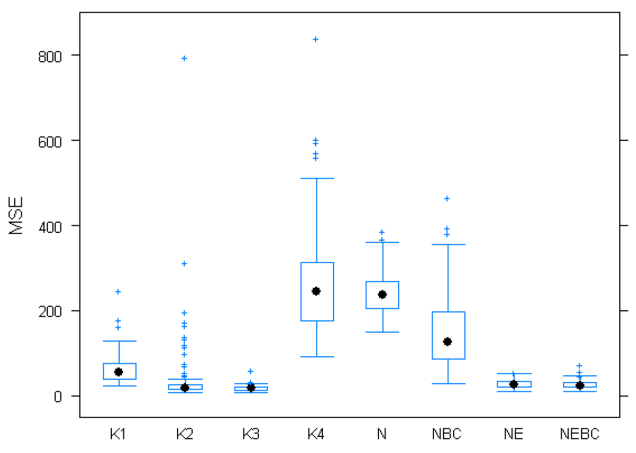


δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = -0.5 | δ_{θ_1} = 0 | δ_{θ_2} = 0.5
MSE
600
500
400
300
200
100
0
K1
K2
K3
K4
N
NBC
NE
NEBC

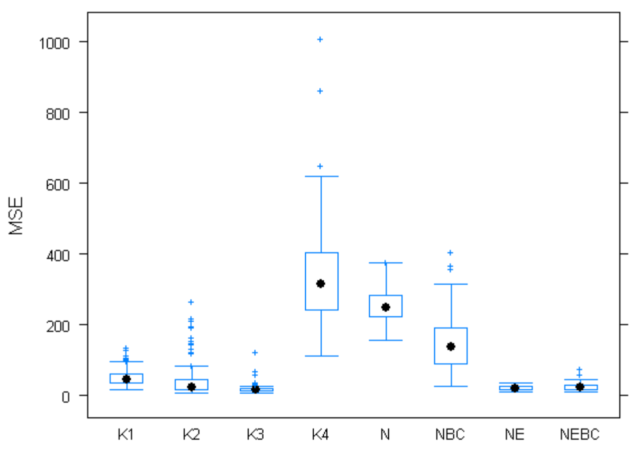


δ_{ϕ_1} = -0.5 | δ_{ϕ_2} = -0.5 | δ_{θ_1} = 0.5 | δ_{θ_2} = -0.5
MSE
500
400
300
200
100
0
K1
K2
K3
K4
N
NBC
NE
NEBC

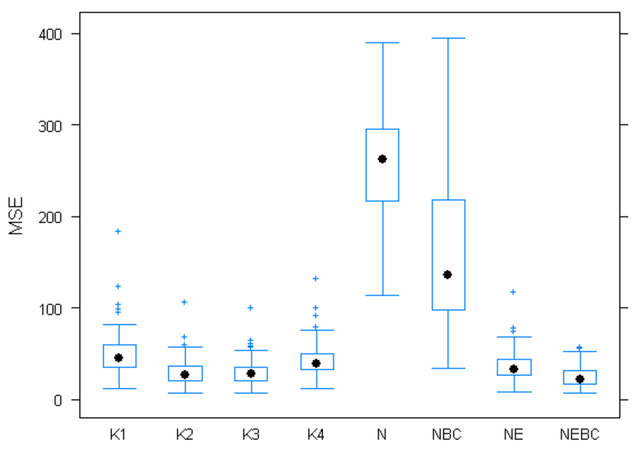
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$



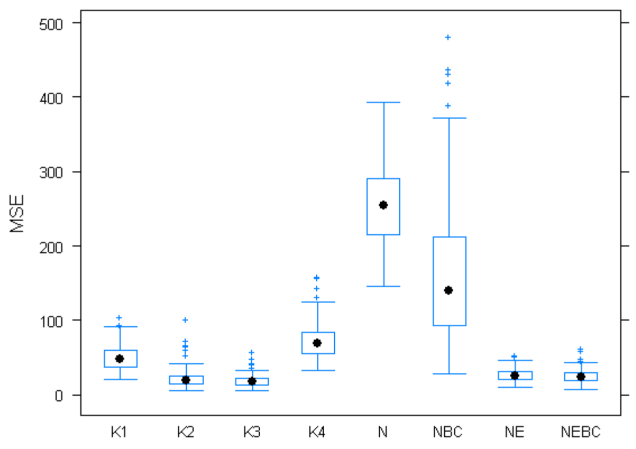
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$



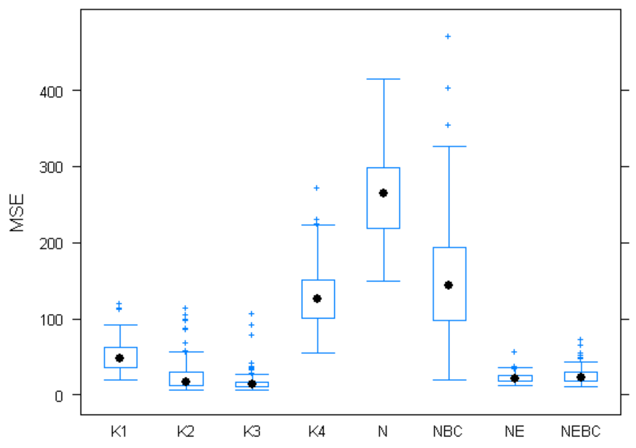
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$



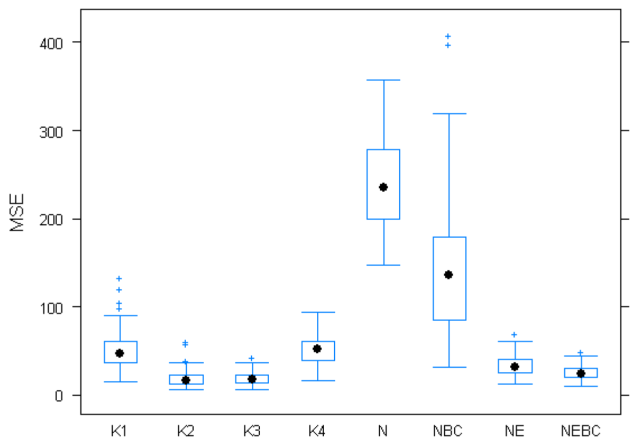
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$



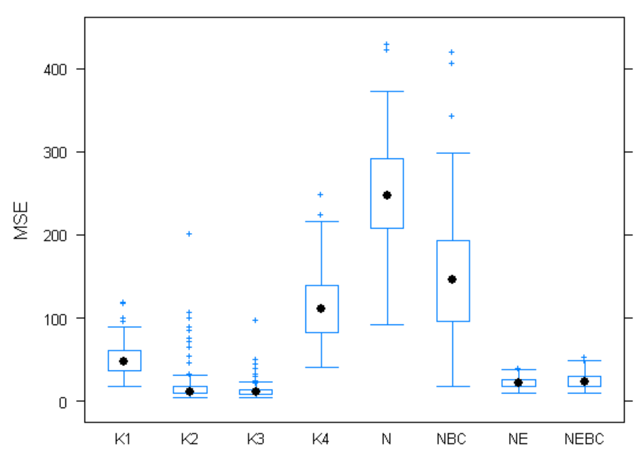
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$



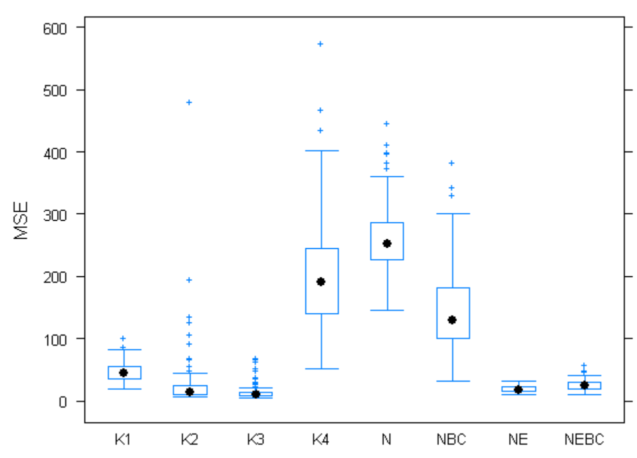
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$



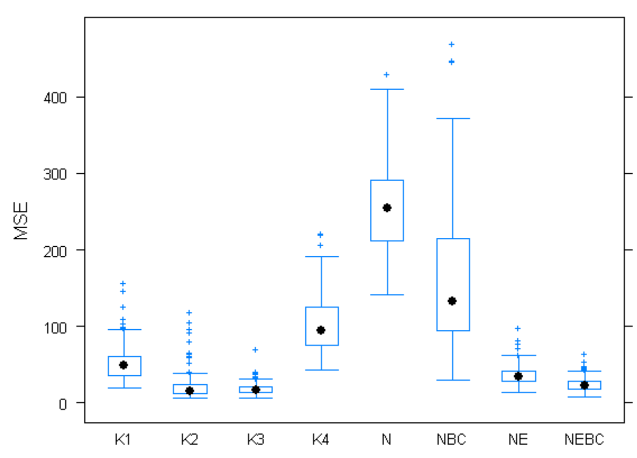
$\delta_{\phi_1} = -0.5 \mid \delta_{\phi_2} = 0 \mid \delta_{\theta_1} = 0 \mid \delta_{\theta_2} = 0$



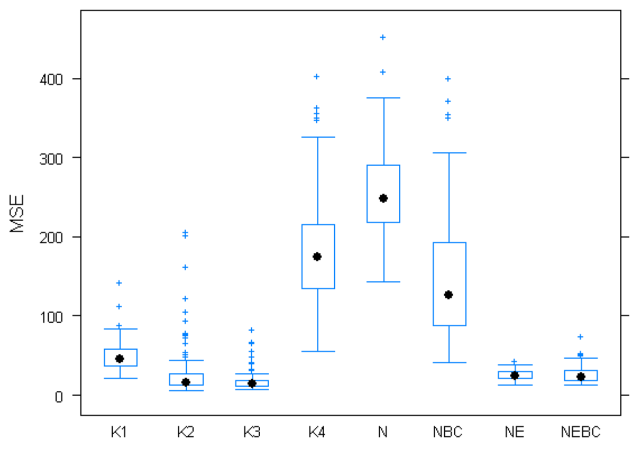
$\delta_{\phi_1} = -0.5 \mid \delta_{\phi_2} = 0 \mid \delta_{\theta_1} = 0 \mid \delta_{\theta_2} = 0.5$



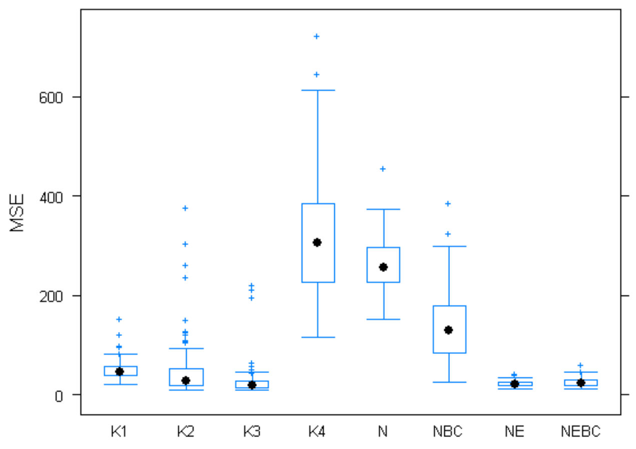
$\delta_{\phi_1} = -0.5 \mid \delta_{\phi_2} = 0 \mid \delta_{\theta_1} = 0.5 \mid \delta_{\theta_2} = -0.5$



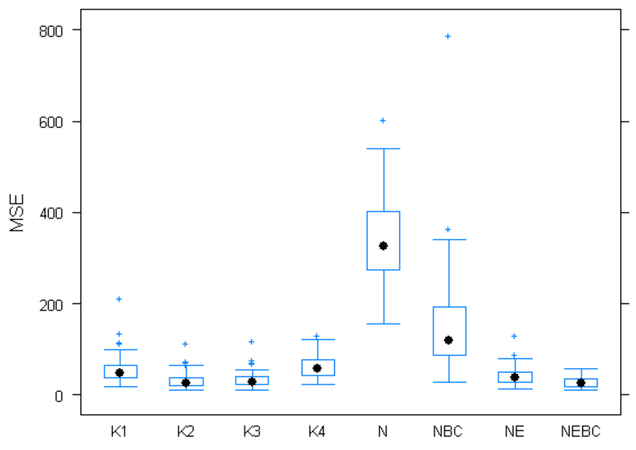
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$

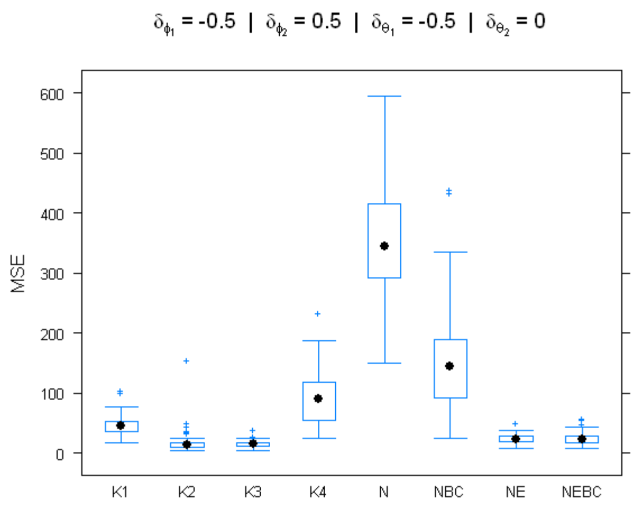


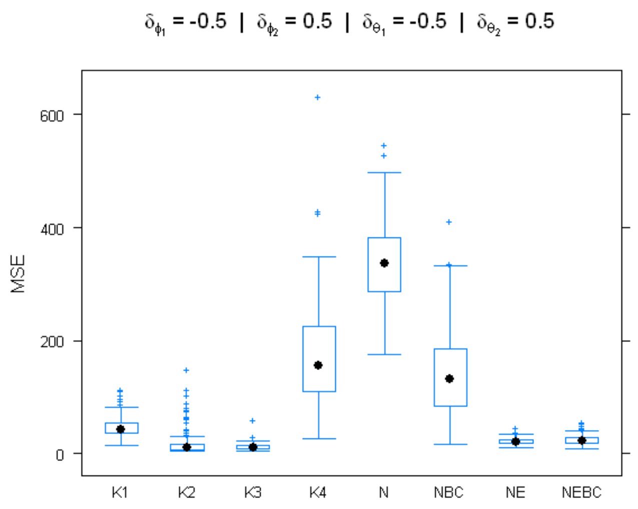
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$

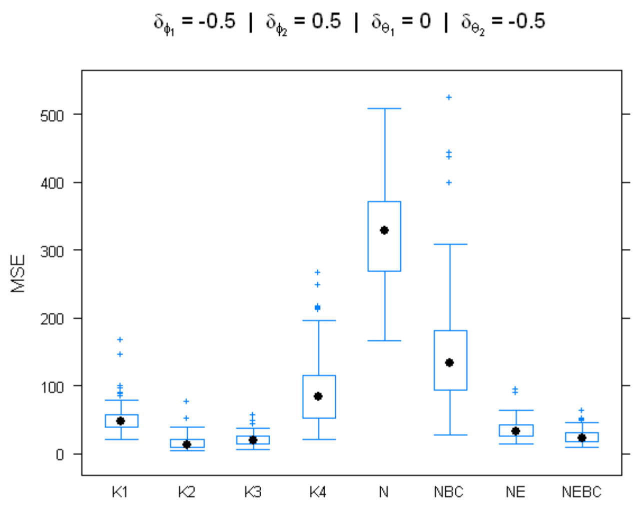


$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$

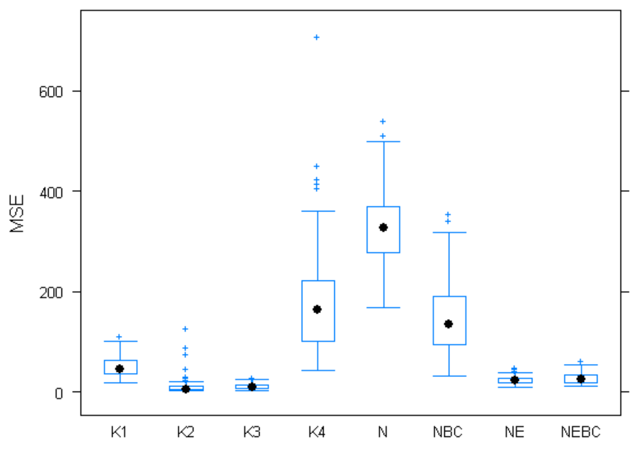



$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC

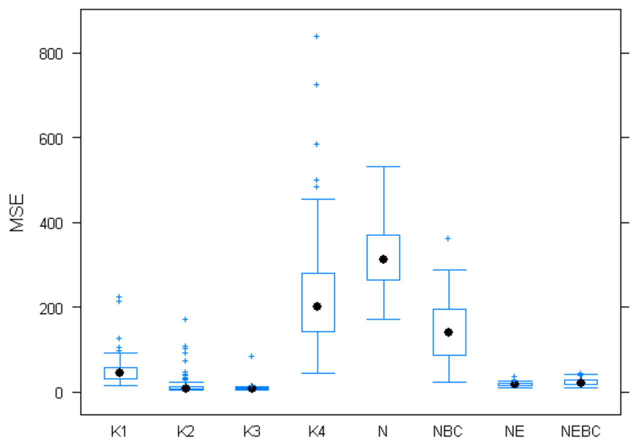

$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
500
K1
K2
K3
K4
N
NBC
NE
NEBC

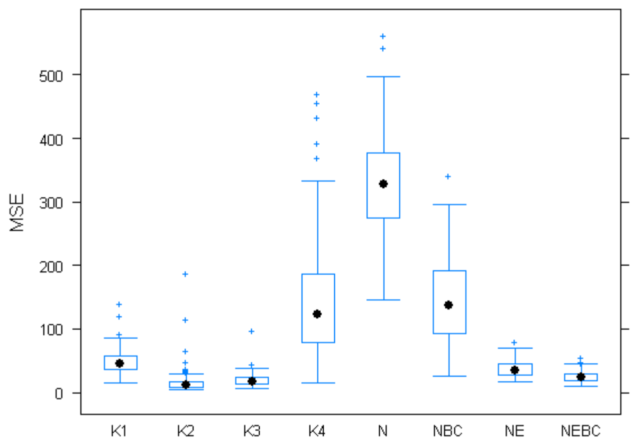
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0$



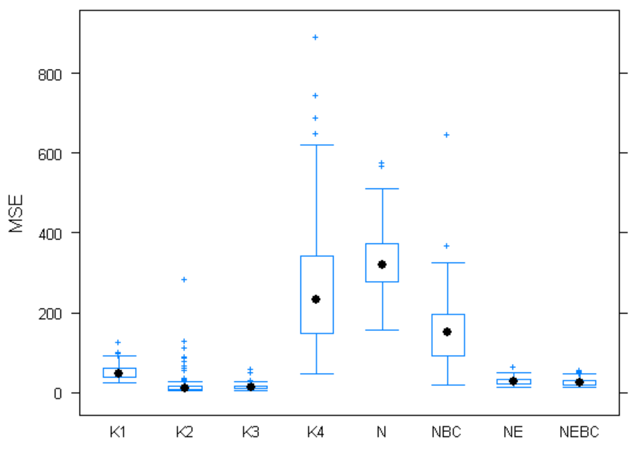
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$



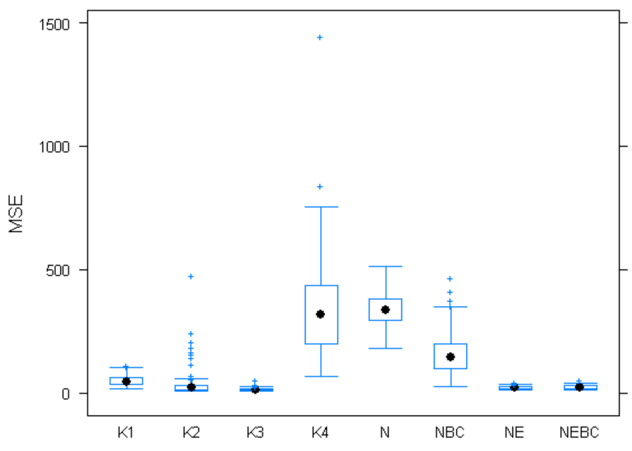
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$



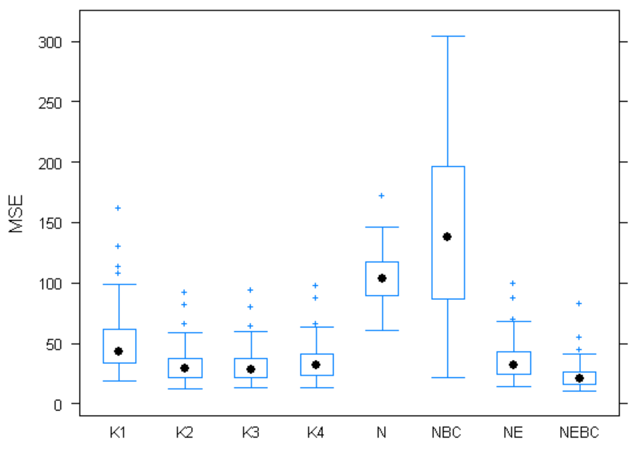
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$



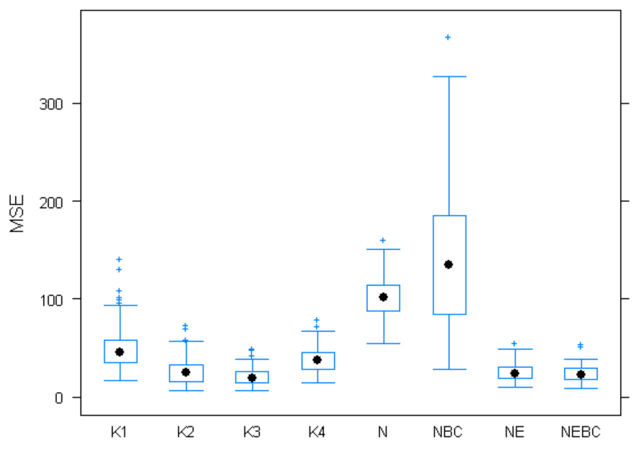
$\delta_{\phi_1} = -0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$



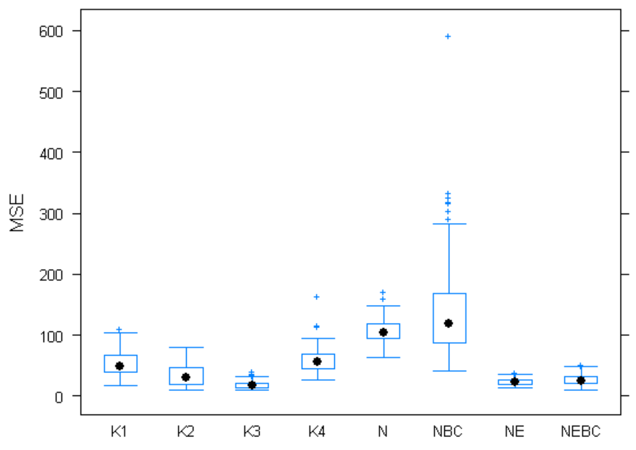
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$



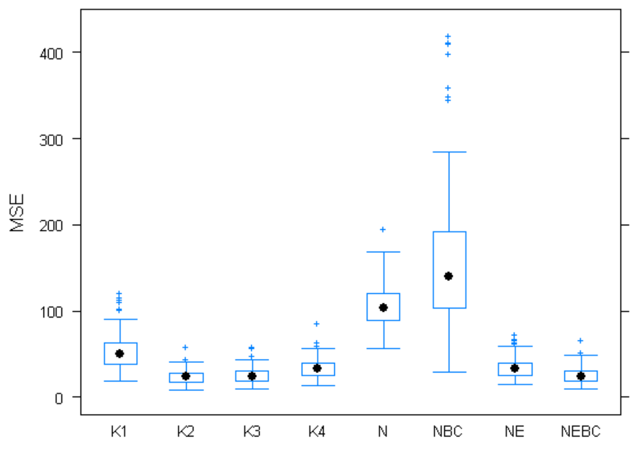
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$

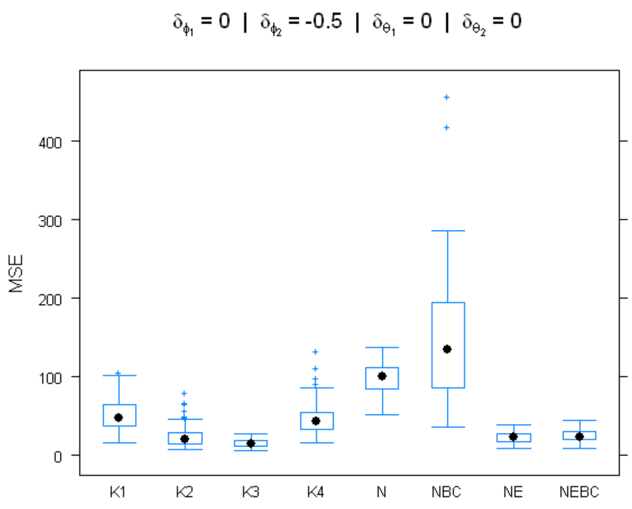


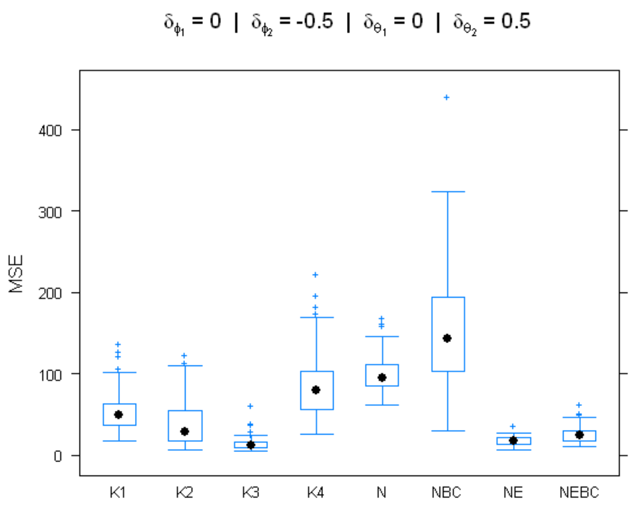
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$

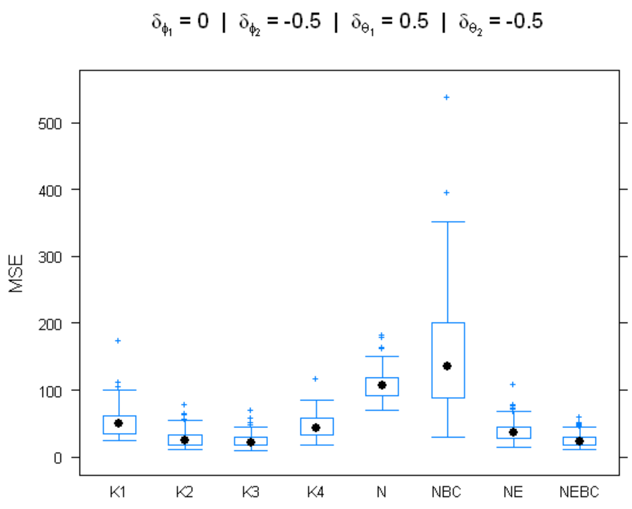


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$

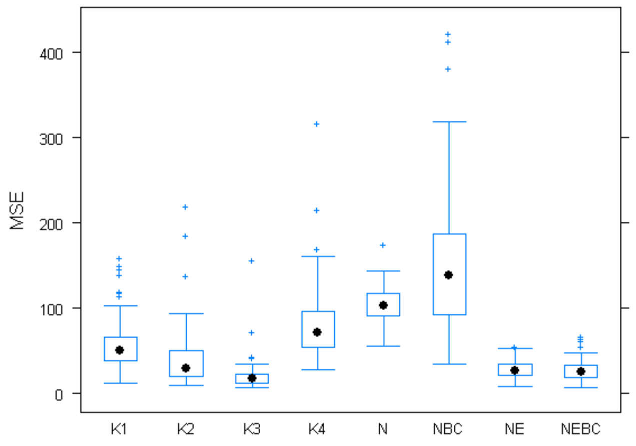



$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

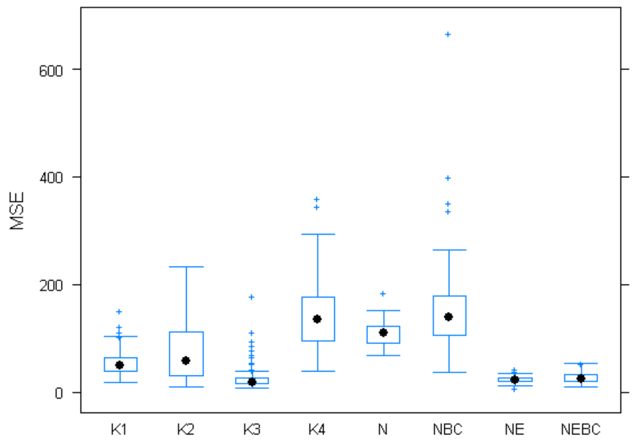

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
500
K1
K2
K3
K4
N
NBC
NE
NEBC

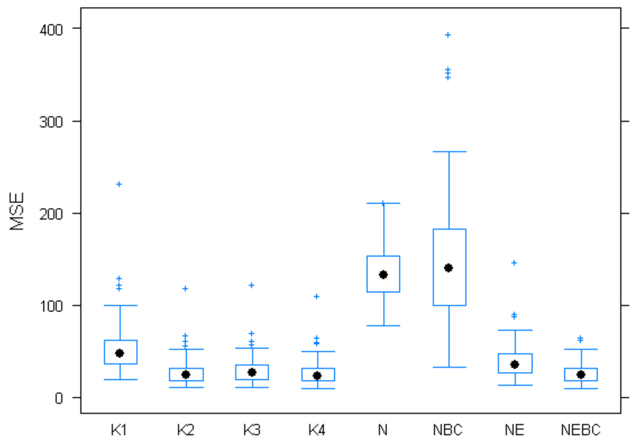
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$

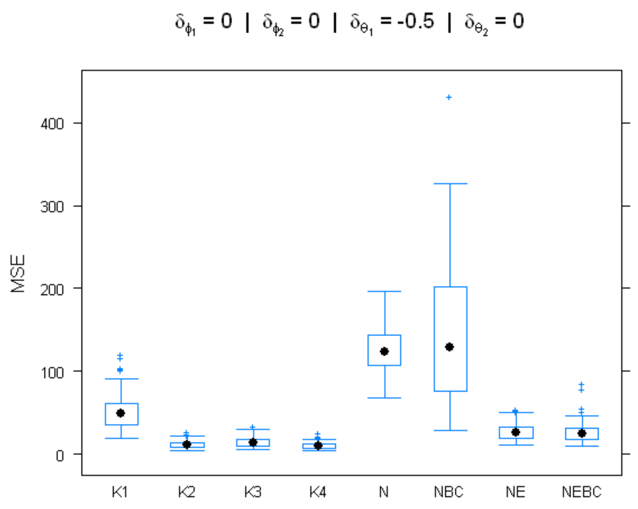


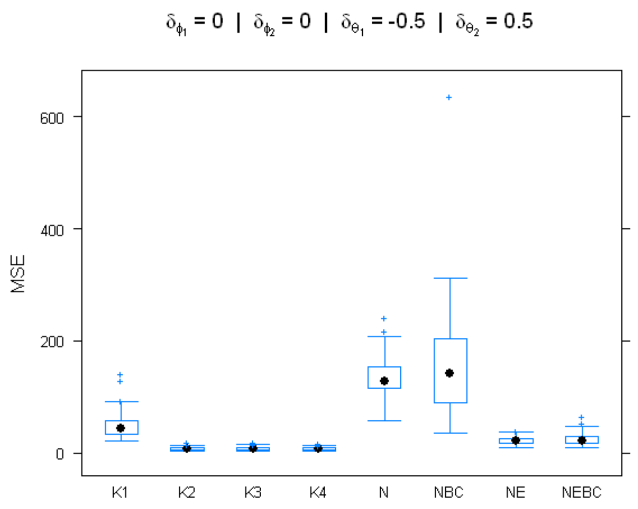
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$

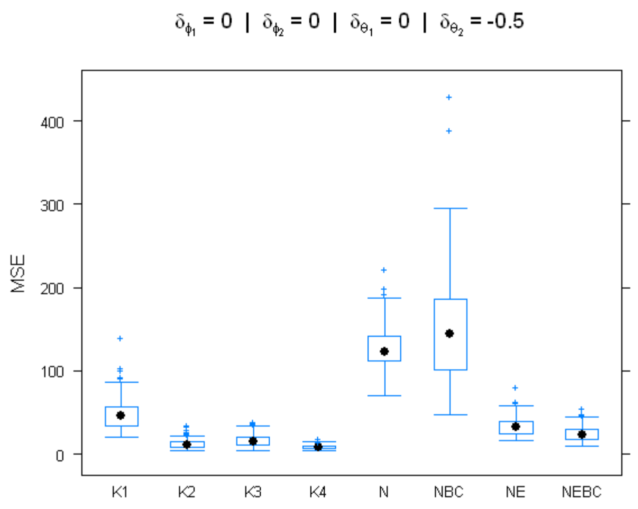


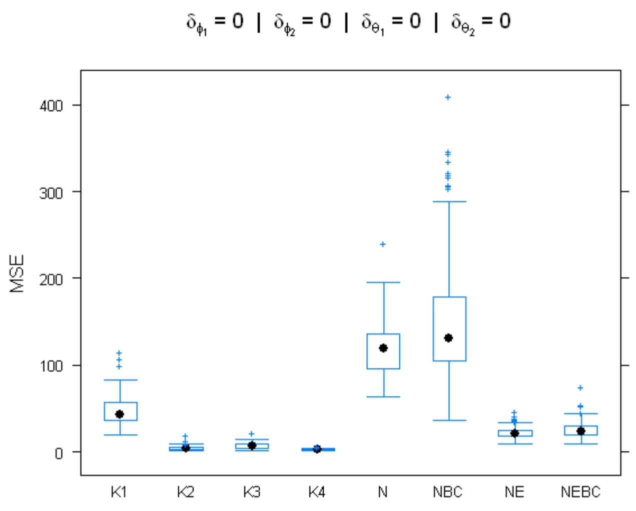
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$

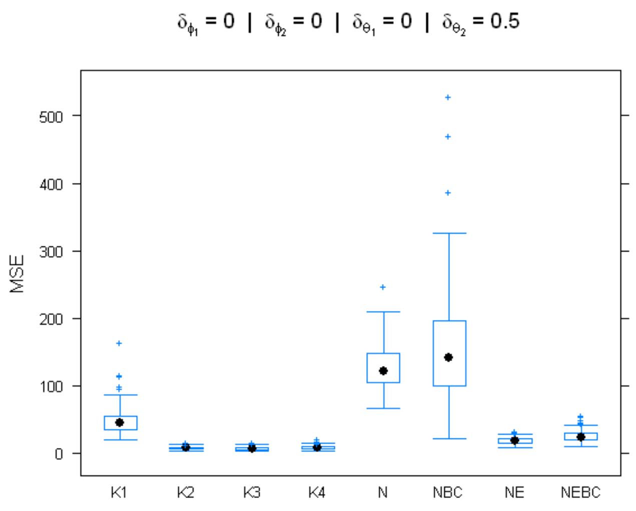


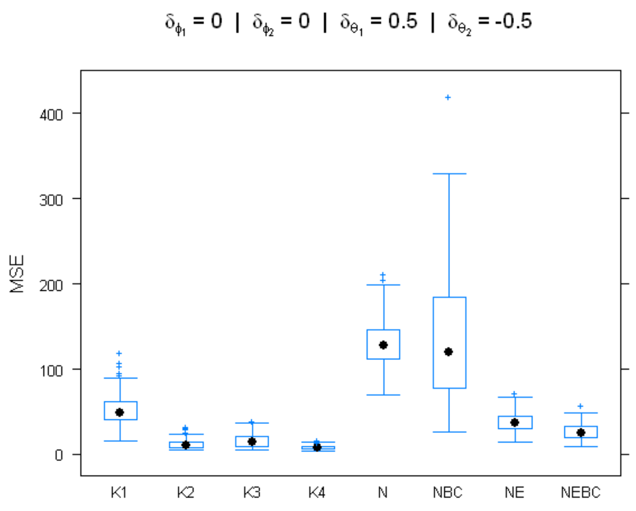

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

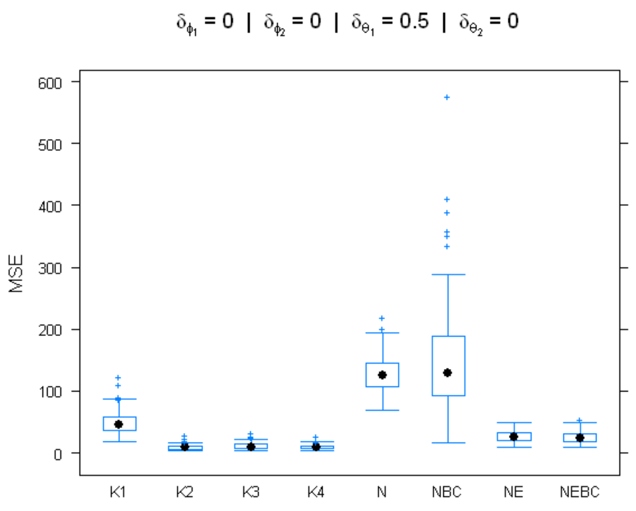

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

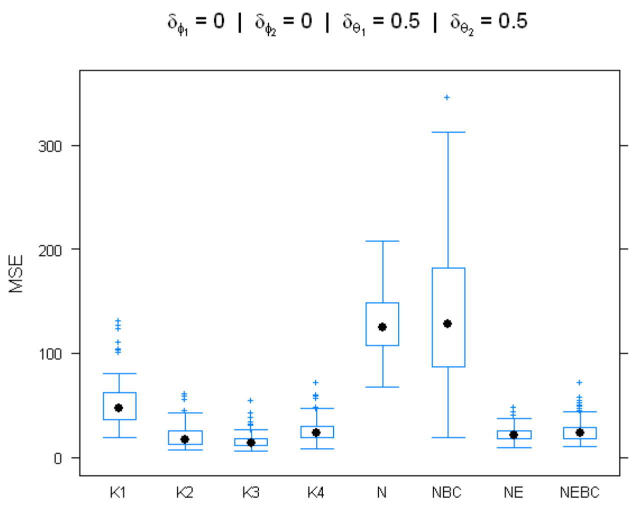

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0$
MSE
400
300
200
100
0
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$
MSE
500
400
300
200
100
0
K1
K2
K3
K4
N
NBC
NE
NEBC

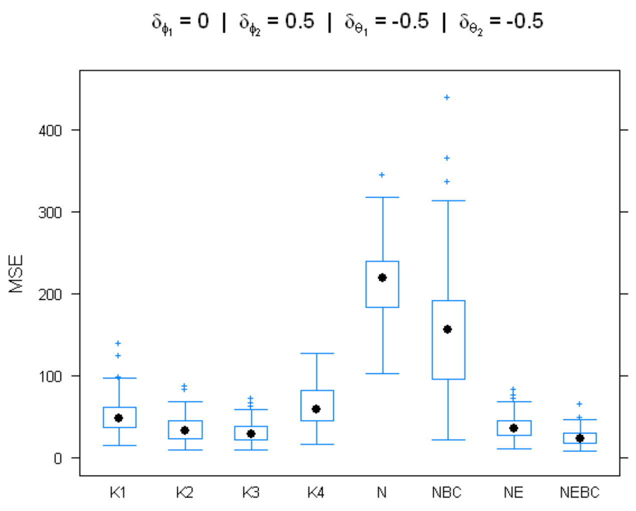

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$
MSE
400
300
200
100
0
K1
K2
K3
K4
N
NBC
NE
NEBC



$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC

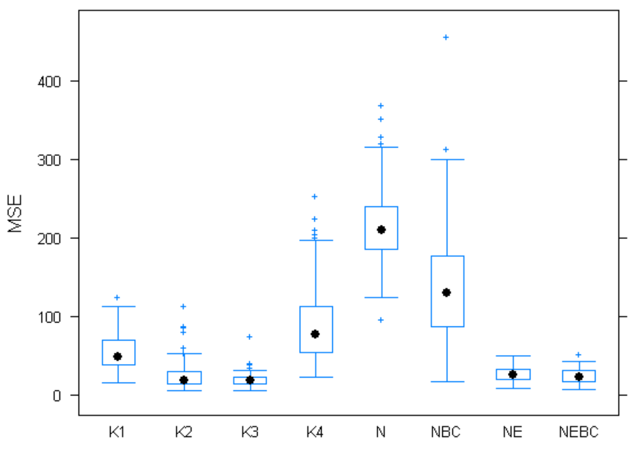


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
100
200
300
K1
K2
K3
K4
N
NBC
NE
NEBC

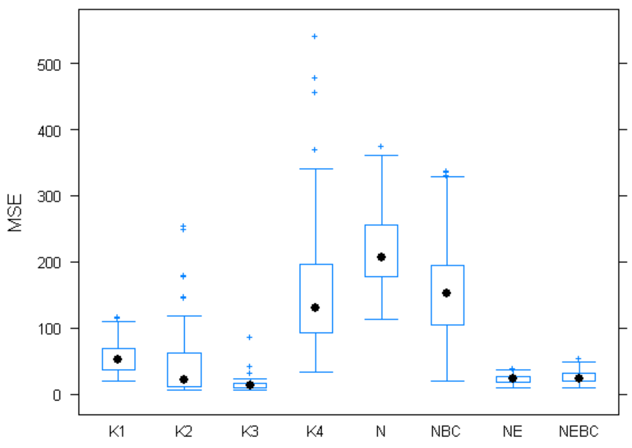


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

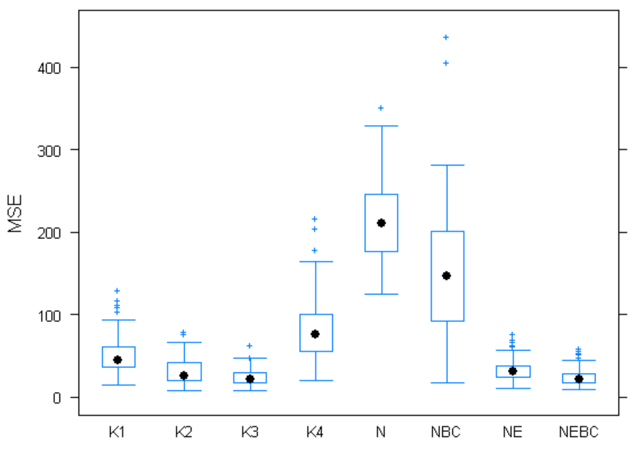
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$

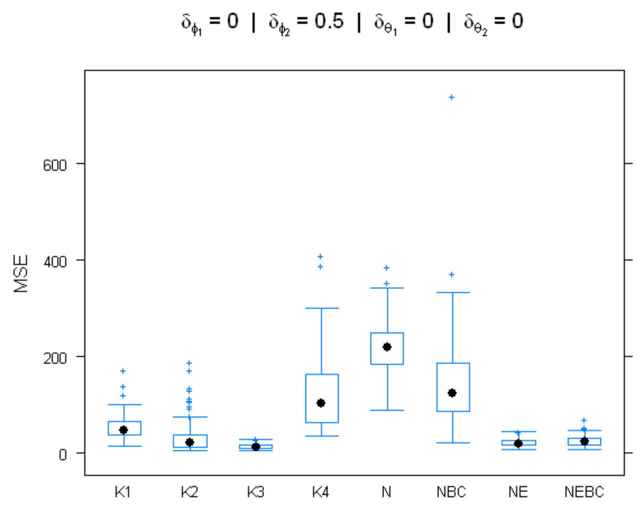


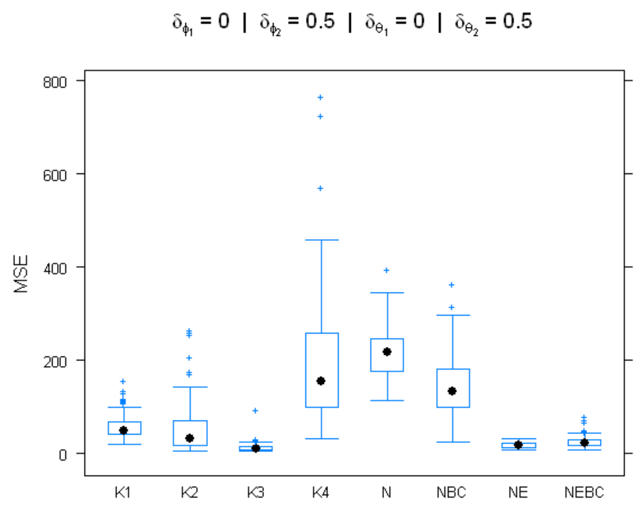
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$

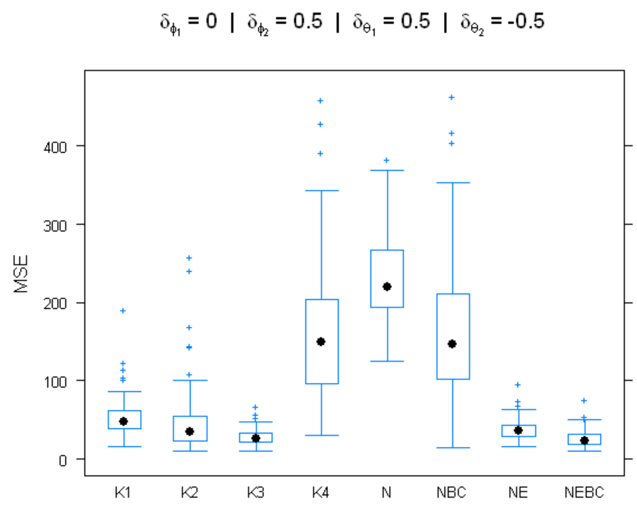


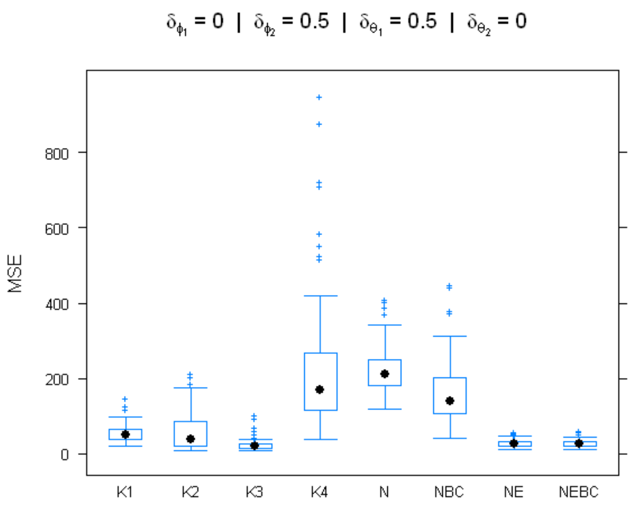
$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$

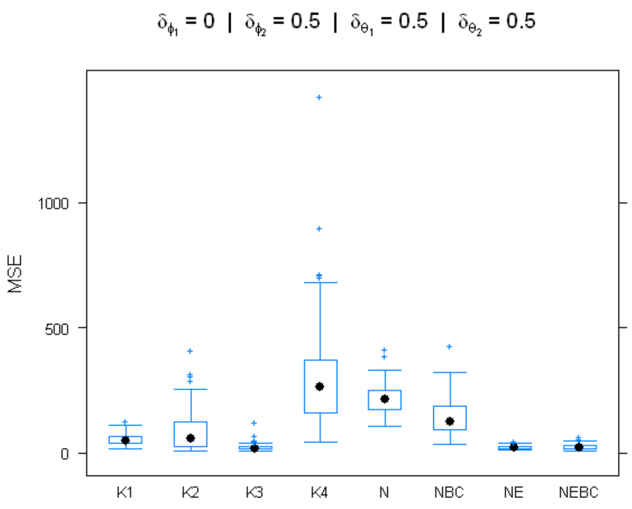


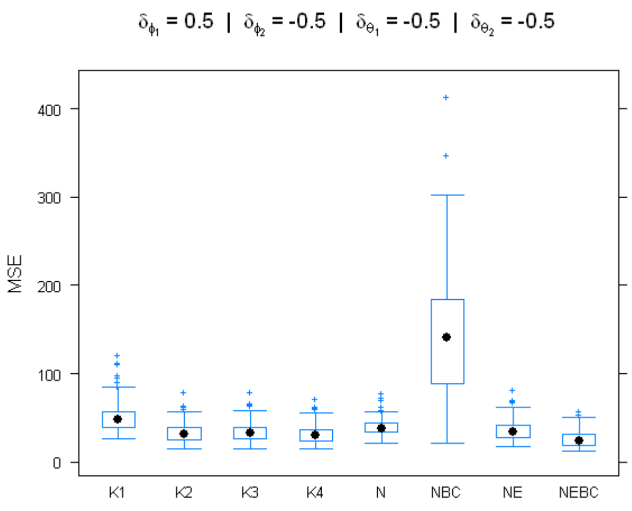

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

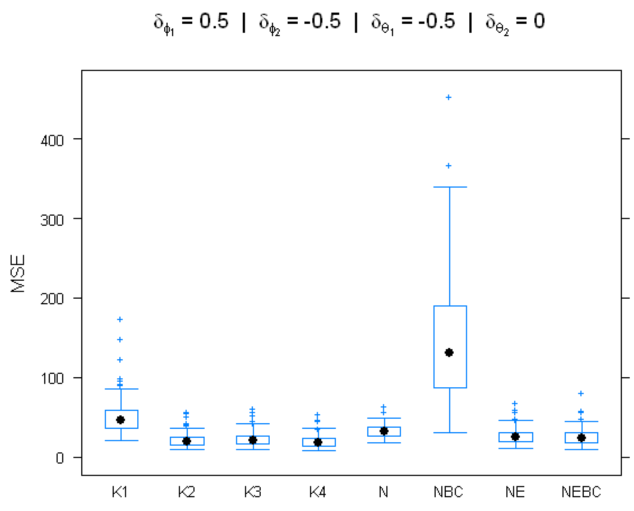

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC

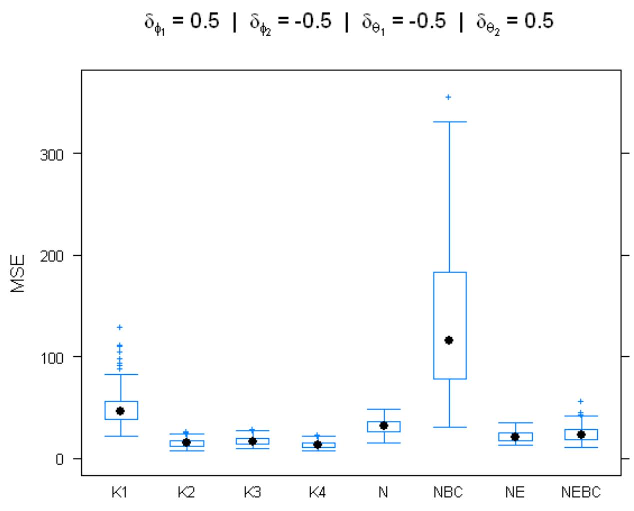

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

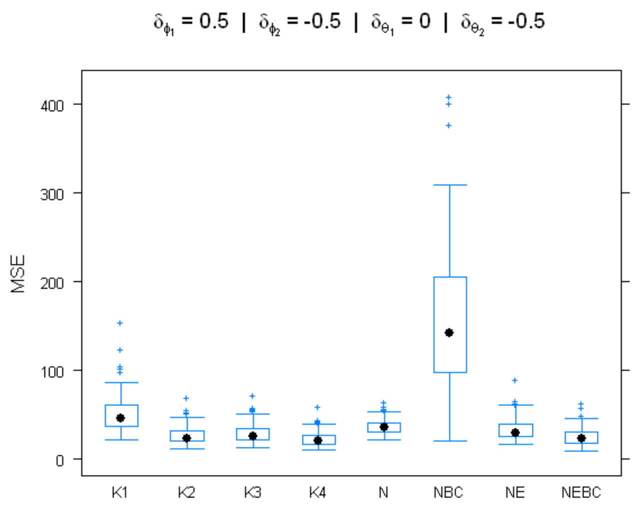

$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$
MSE
0
200
400
600
800
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
500
1000
K1
K2
K3
K4
N
NBC
NE
NEBC

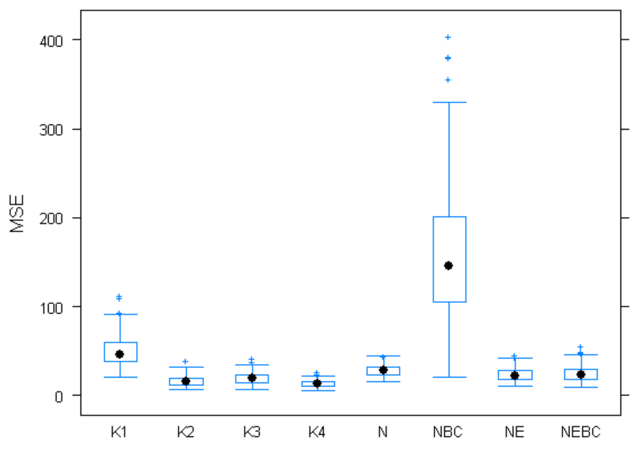

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC


δ_{φ1} = 0.5 | δ_{φ2} = -0.5 | δ_{θ1} = -0.5 | δ_{θ2} = 0
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

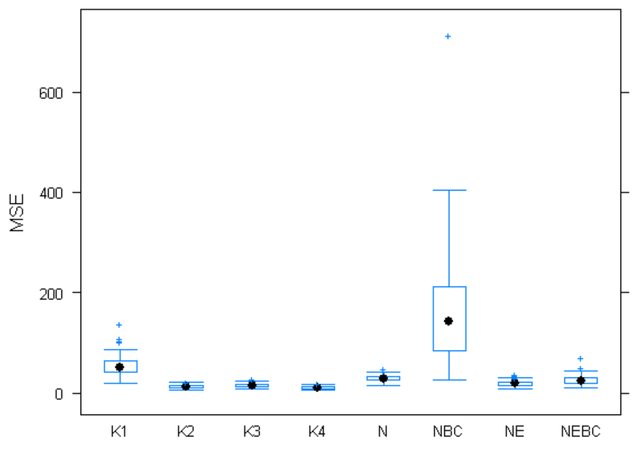

δ_{φ1} = 0.5 | δ_{φ2} = -0.5 | δ_{θ1} = -0.5 | δ_{θ2} = 0.5
MSE
0
100
200
300
K1
K2
K3
K4
N
NBC
NE
NEBC


δ_{φ1} = 0.5 | δ_{φ2} = -0.5 | δ_{θ1} = 0 | δ_{θ2} = -0.5
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

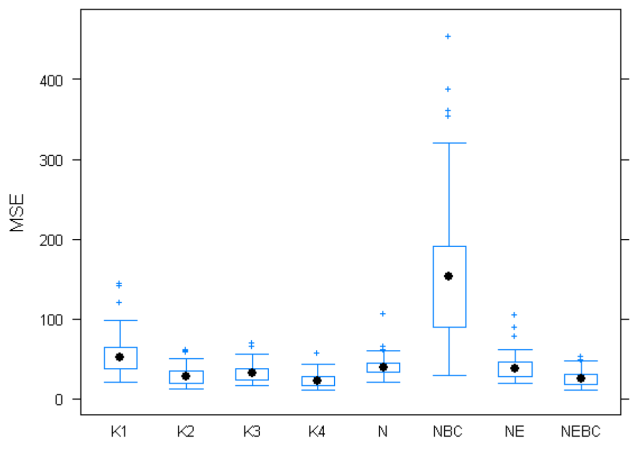
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0$



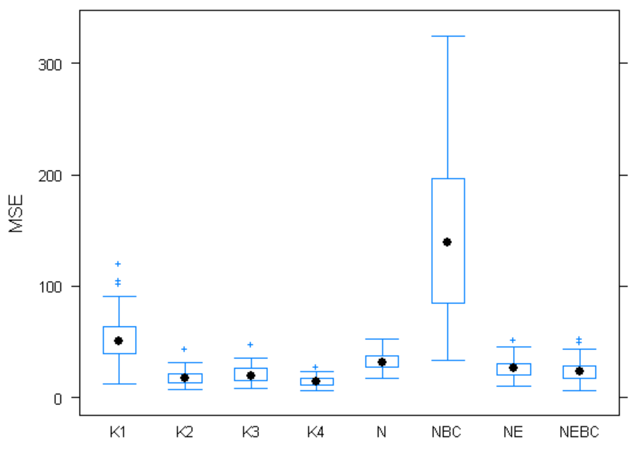
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = 0.5$



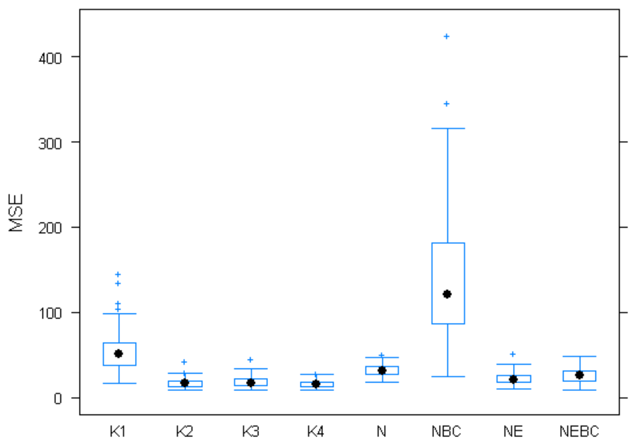
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = -0.5$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = -0.5$



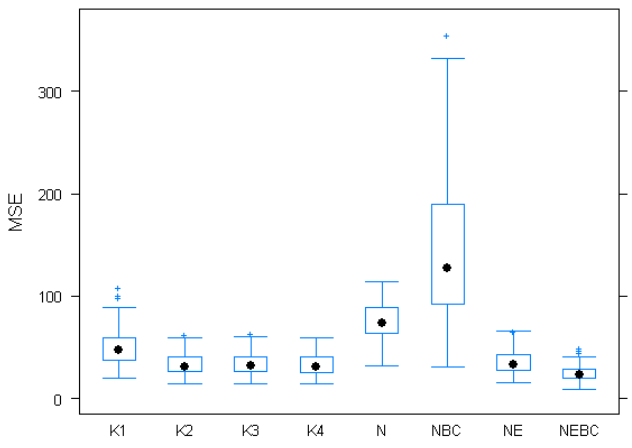
$\delta_{\phi_1} = 0.5 \;|\; \delta_{\phi_2} = -0.5 \;|\; \delta_{\Theta_1} = 0.5 \;|\; \delta_{\Theta_2} = 0$

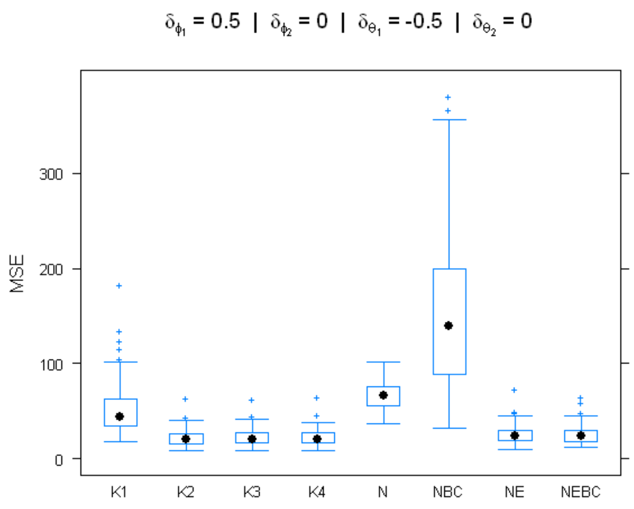


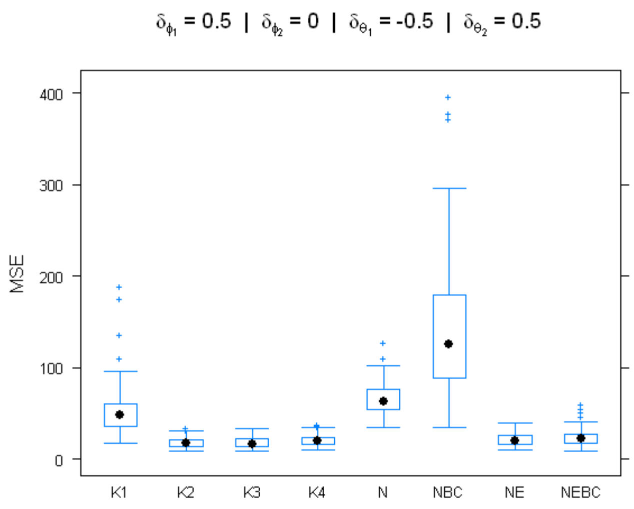
$\delta_{\phi_1} = 0.5 \;|\; \delta_{\phi_2} = -0.5 \;|\; \delta_{\Theta_1} = 0.5 \;|\; \delta_{\Theta_2} = 0.5$

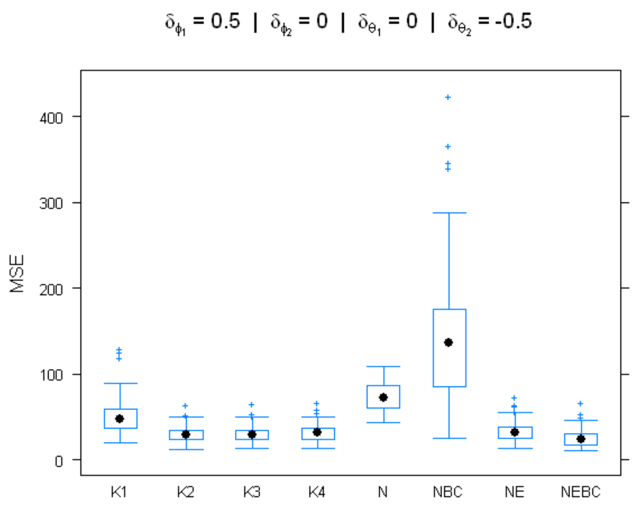


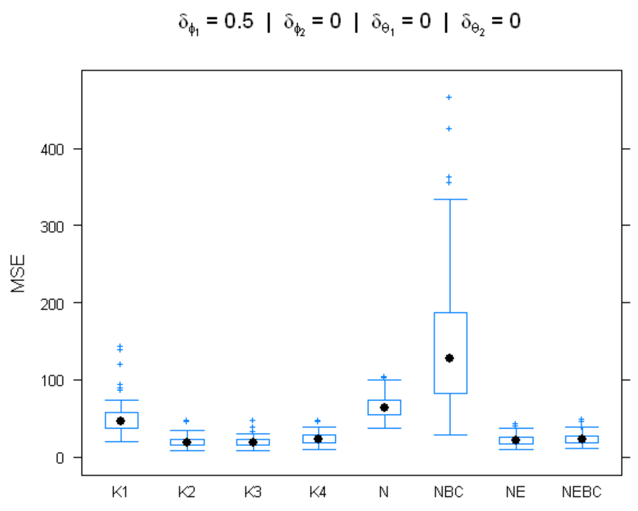
$\delta_{\phi_1} = 0.5 \;|\; \delta_{\phi_2} = 0 \;|\; \delta_{\Theta_1} = -0.5 \;|\; \delta_{\Theta_2} = -0.5$

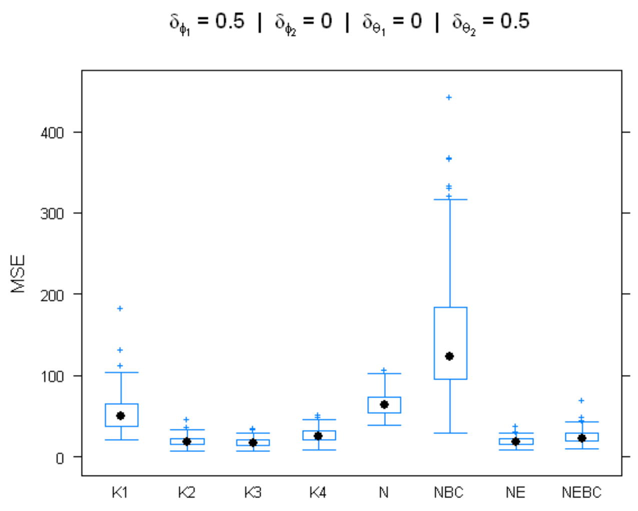


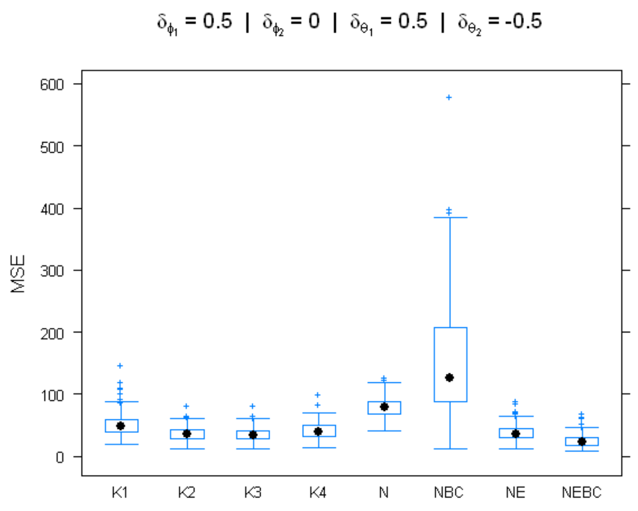

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
K1
K2
K3
K4
N
NBC
NE
NEBC

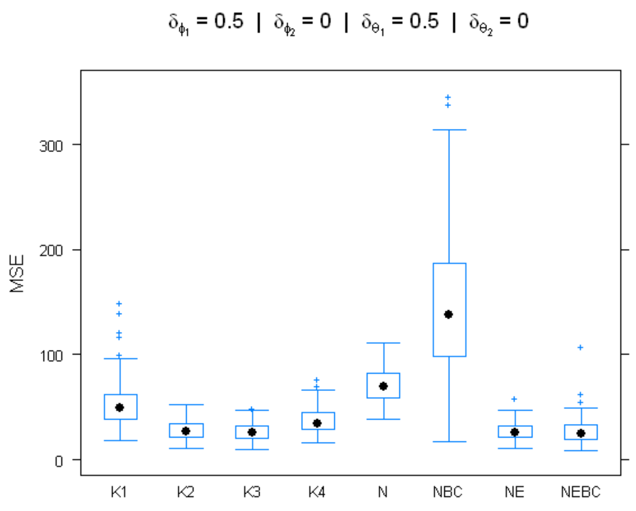

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

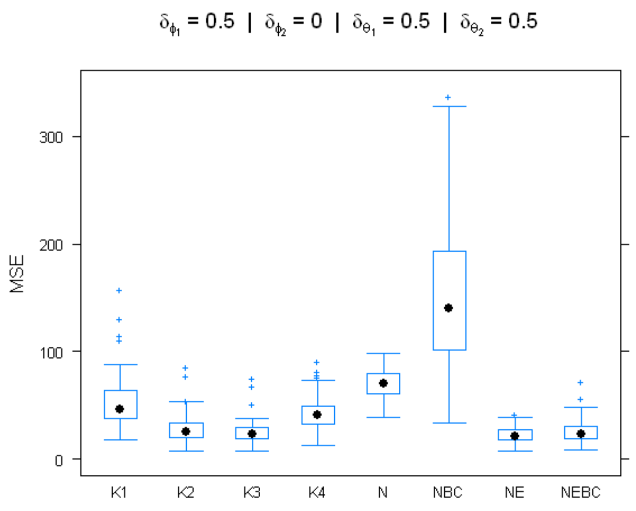

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

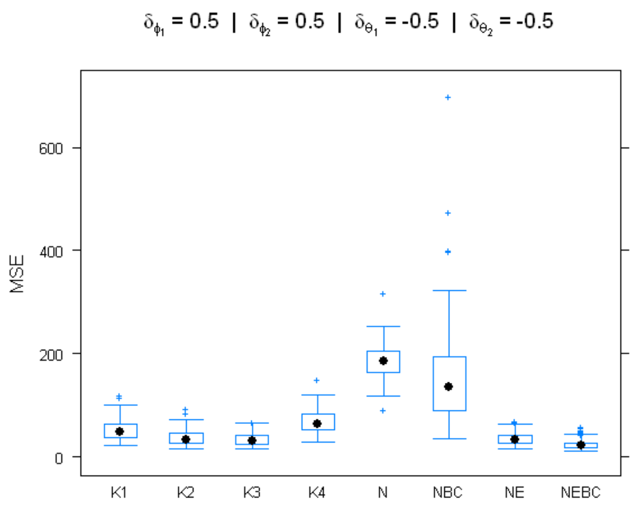

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\Theta_1} = 0$ | $\delta_{\Theta_2} = 0$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

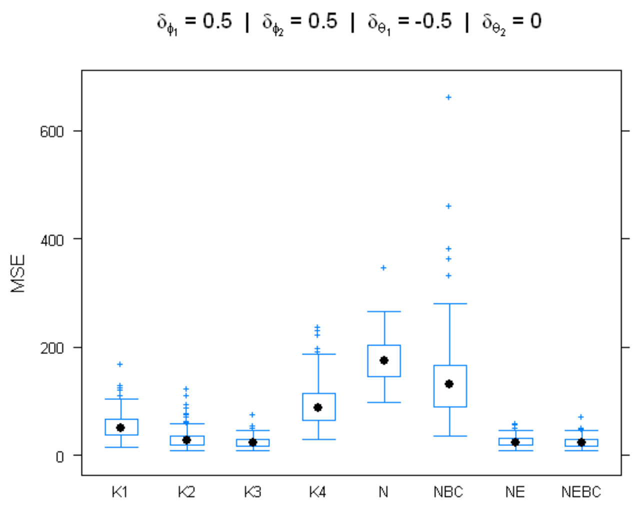

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\Theta_1} = 0$ | $\delta_{\Theta_2} = 0.5$
MSE
0
100
200
300
400
K1
K2
K3
K4
N
NBC
NE
NEBC

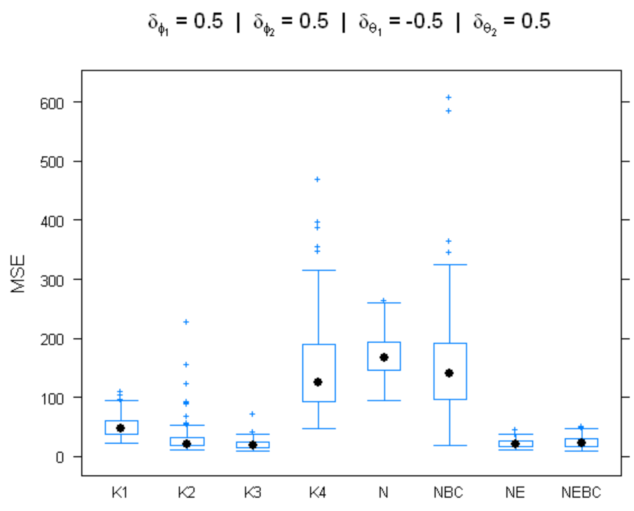

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\Theta_1} = 0.5$ | $\delta_{\Theta_2} = -0.5$
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC

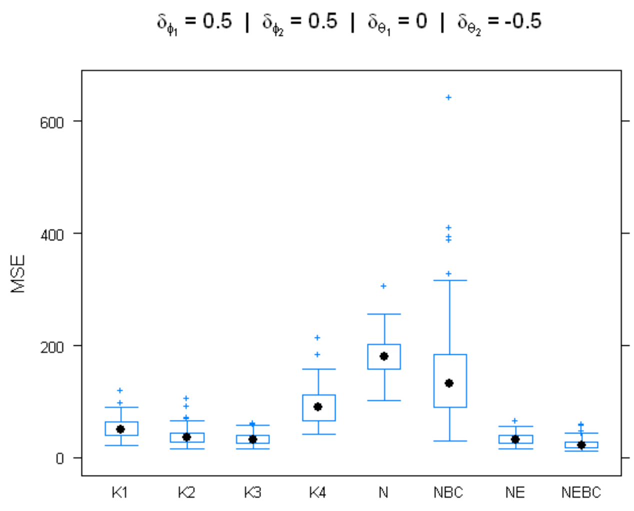

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0$
MSE
0
100
200
300
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0$ | $\delta_{\theta_1} = 0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
100
200
300
K1
K2
K3
K4
N
NBC
NE
NEBC

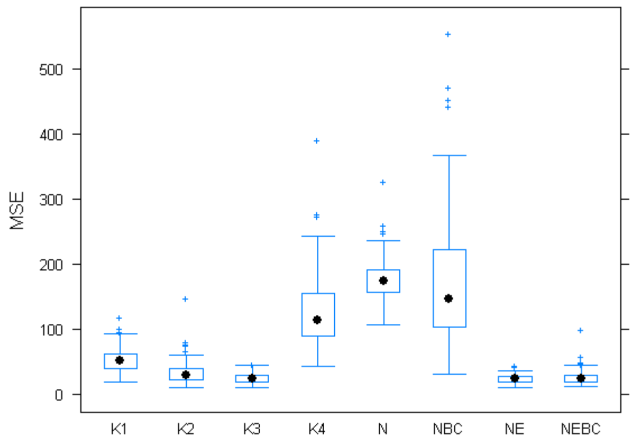

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = -0.5$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

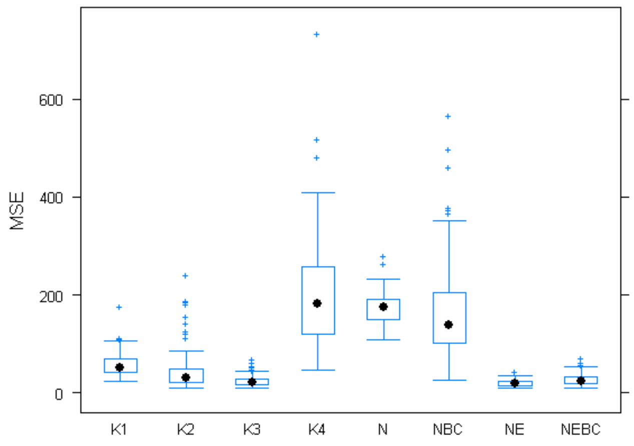

$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = -0.5$ | $\delta_{\theta_2} = 0.5$
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC


$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\theta_1} = 0$ | $\delta_{\theta_2} = -0.5$
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC

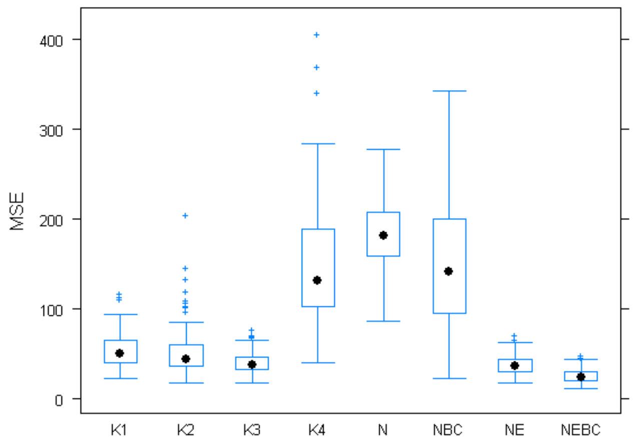
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = 0$ | $\delta_{\Theta_2} = 0$

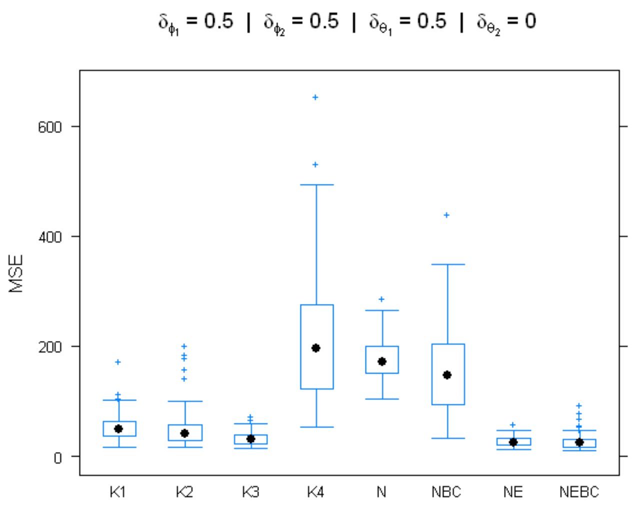


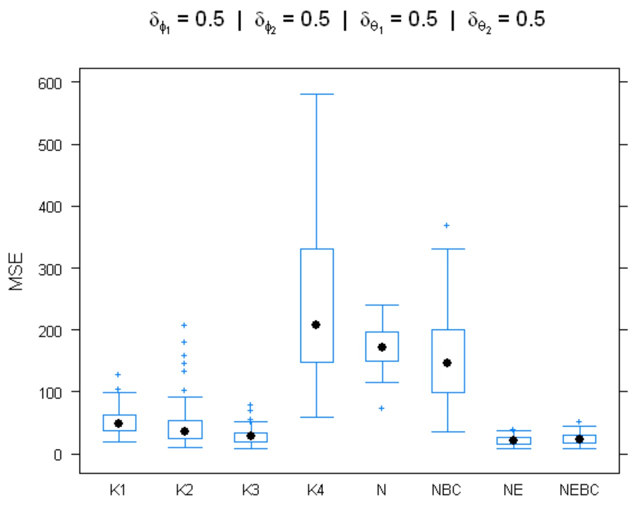
$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = 0$ | $\delta_{\Theta_2} = 0.5$



$\delta_{\phi_1} = 0.5$ | $\delta_{\phi_2} = 0.5$ | $\delta_{\Theta_1} = 0.5$ | $\delta_{\Theta_2} = -0.5$




δ_{ϕ_1} = 0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = 0.5 | δ_{θ_2} = 0
MSE
0
200
400
600
K1
K2
K3
K4
N
NBC
NE
NEBC


δ_{ϕ_1} = 0.5 | δ_{ϕ_2} = 0.5 | δ_{θ_1} = 0.5 | δ_{θ_2} = 0.5
MSE
0
100
200
300
400
500
600
K1
K2
K3
K4
N
NBC
NE
NEBC

B.1 Matlab

B.1.1 State Estimate Functions

Krein Space Kalman Filter

```
function [xn, Pn] = state_krein(xo, yn, Po_krein, Ad, C, K, Veta, Veps, G1, G2)

    [r1, c1] = size(G1);
    Ct       = [C; K];
    [aa, tt] = size(Veta);
    [r2, c2] = size(C);

    VEps_e = [eye(c1) zeros(c1,r1); zeros(r1,c1)  Veps];
    Vetat  = [Veta zeros(aa,r2); zeros(r2,aa)  eye(r2)];
    VEta_e = Ct*Po_krein*Ct' + Vetat;
    G      = [G1 G2];

    L  = Ad*Po_krein*Ct'*inv(VEta_e);
    yk = [(yn - C*xo); (-K*xo)];
    xn = Ad*xo + L*yk;

    Pnn = Po_krein - Po_krein*Ct'*inv(VEta_e)*Ct*Po_krein;
    Pn  = Ad*Pnn*Ad' + G*VEps_e*G';

return;
```

Adjusted Krein Space Kalman Filter

```
function [xn, Pn] = state_krein_IR(xo, yn, Po_krein, Ad, C, K, Veta, Veps, G1, G2, un, Bd)

    [r1, c1] = size(G1);
    Ct       = [C; K];
    [aa, tt] = size(Veta);
    [r2, c2] = size(K);

    VEps_e = [[eye(c1)       zeros(c1,r1)]; ...
              [zeros(r1,c1)  Veps]];
    Vetat  = [[Veta          zeros(aa,r2)]; ...
              [zeros(r2,aa)  eye(r2)]];
    VEta_e = Ct*Po_krein*Ct' + Vetat;
    G      = [G1 G2];

    L  = Ad*Po_krein*Ct'*inv(VEta_e);
    yk = [(yn - C*xo); (-K*xo)];
    xn = Ad*xo + Bd*un + L*yk;

    Pnn = Po_krein - Po_krein*Ct'*inv(VEta_e)*Ct*Po_krein;
    Pn  = Ad*Pnn*Ad' + G*VEps_e*G';

return;
```

Classic Kalman Filter

```
function [xx, P_new] = State_Kalman(A, B, G, old_x, P_old, u, C, y, V_eps, V_eta)

    Ln = A*P_old*C'*inv(V_eta);
    xx = A*old_x + B*u + Ln*(y - C*old_x);
    P = A*P_old*A' + G*V_eps*G';
    %P_new = pinv(pinv(M) + C'*inv(V_eta)*C);
    P_new = P - P*C'*inv(C*P*C' + V_eta)*C*P;

return;
```

Naive Filter

```
function [state] = State_Naive(u, z, Tu, Tz, Ad, Bd)

    xx = Tz*z + Tu*u;
    state = Ad*xx + Bd*u(1);

return;
```

Bias Corrected Naive Filter

```
function [stateBC] = State_NaiveBC(u, z, Tu, Tz, Ad, Bd, Adelt, Bdelt, Tz_del, Tu_del)

    xx = Tz*z + Tu*u;
    xxb = Tz_del*z + Tu_del*u;
    stateB = Ad*xx + Bd*u(1);
    Bias = Adelt*xxb + Bdelt*u(1);
    stateBC = stateB - Bias;

return;
```

EWMA Naive Filter

```
function [state, znew] = State_NaiveE(u, z, zold, Tu, Tz, Ad, Bd, lambda)

    znew = ewma(z, zold, lambda);
    xx = Tz*znew + Tu*u;
    state = Ad*xx + Bd*u(1);

return;
```

Bias Corrected EWMA Naive Filter

```
function [stateBC, znew] = State_NaiveEBC(u, z, zold, Tu, Tz, Ad, Bd, Adelt, Bdelt, Tz_del, Tu_del, lambda)

    znew = ewma(z, zold, lambda);
    xx = Tz*znew + Tu*u;
    xxB = Tz_del*znew + Tu_del*u;
    stateB = Ad*xx + Bd*u(1);
    Bias = Adelt*xxB + Bdelt*u(1);
    stateBC = stateB - Bias;

return;
```

EWMA

```
function [new, lam] = ewma(zn, old, lam)% phi, theta)

    new = lam * zn + (1 - lam) * old;

return;
```

B.1.2 Simulation Code

Overall Simulation

```
% Call file names in for the 81 different files.

filen = textread('C:\My Files\Thesis\Filenames\FilenamesMSE52noP.txt', '%q');

% The next piece creates the indexing matrix to index the MSE file names in
% correspondence with the MSE values created.

Filen = zeros(3,3,3,3);
L = 0;
```

```
for k = 1:3
for m = 1:3
for g = 1:3
for b = 1:3
    L = L + 1;
    Filen(g,b,m,k) = L;
end
end
end
end

% This demonstrates that the indexing works.

for b = 1:3
for g = 1:3
for m = 1:3
for k = 1:3
    filen(Filen(m,k,g,b))
end
end
end
end

% Here is the code for the overall simulation control program, which utilizes base_sim as the
% workhorse function. Then each matrix is written out to a file.

% For a trace of length 52.

reps = 100;
delta = [-0.5 0 0.5];
MSE = zeros(reps, 8);
for b = 1:3
for g = 1:3
for m = 1:3
for k = 1:3
for j = 1:reps
    MSE(j,:) = base_sim(52, 50, -0.95, 0.6, 0.6, 0.9, delta(b), delta(g), delta(m),
     delta(k), 1, 10, 3, 0.00001, 3)';
end
fid = fopen(char(filen(Filen(m,k,g,b))),'w');
fprintf(fid, '%15.6f %15.6f %15.6f %15.6f %15.6f %15.6f %15.6f %15.6f\n', MSE);
fclose(fid);
end
end
end
end

% For a trace of length 12.

filen = textread('C:\My Files\Thesis\Filenames\FilenamesMSE12noP.txt', '%q');
reps = 100;
delta = [-0.5 0 0.5];
MSE = zeros(reps, 8);
for b = 1:3
for g = 1:3
for m = 1:3
for k = 1:3
for j = 1:reps
    MSE(j,:) = base_sim(12, 10, -0.95, 0.6, 0.6, 0.9, delta(b), delta(g), delta(m), delta(k),
     1, 10, 3, 0.00001, 3)';
end
fid = fopen(char(filen(Filen(m,k,g,b))),'w');
fprintf(fid, '%15.6f %15.6f %15.6f %15.6f %15.6f %15.6f %15.6f %15.6f\n', MSE);
fclose(fid);
end
end
end
end
```

Workhorse Function

```
function [mse] = base_sim(n, ind, phi1, phi2, tht1, tht2, dp1, dp2, dt1, dt2, sde, sdn, sdu,
lambda, outlr, plot)

    Po_krein = eye(2);
    Po_krein2 = eye(2);
    Po_krein3 = eye(2);
    Po_kalm  = eye(2);

    Ad    = [[-(phi1+dp1) 1]; ...
             [-(phi2+dp2) 0]];

    Bd = [(tht1+dt1); (tht2+dt2)];
    C  = [1 0];
    sdptb = 0.3;
    ptb_phi1 = sdptb*randn(1,1);
    ptb_phi2 = sdptb*randn(1,1);
    ptb_tht1 = sdptb*randn(1,1);
    ptb_tht2 = sdptb*randn(1,1);
    Adelt = [[-(dp1+ptb_phi1) 1]; ...
             [-(dp2+ptb_phi2) 0]];
    Bdelt = [(dt1+ptb_tht1); (dt2+ptb_tht2)];
    K  = [(dp1+ptb_phi1) 0];
    K2 = [ [(dp1+ptb_phi1) 0]; ...
           [(dp2+ptb_phi2) 0] ];

    Veta = sdn^2*eye(1);
    Veps = sde^2*eye(2);
    G1   = [1; 0];
    G1_2 = eye(2);
    G2   =[1 0; 0 0];

    eps  = sde*randn((n+2), 1);
    eta  = sdn*randn((n+2), 1);
    u    = sdu*randn((n+2), 1);

    z = zeros((n+2), 1);
    z(1:2) = 0.1*randn(2,1);
    y = zeros((n+2), 1);

    krein_z   = zeros(2, (n+2));
    krein2_z  = zeros(2, (n+2));
    krein3_z  = zeros(2, (n+2));
    kalman_z  = zeros(2, (n+2));
    naive_z   = zeros(2, (n+2));
    naiveBC_z = zeros(2, (n+2));
    naiveEBC_z = zeros(2, (n+2));
    naiveE_z  = zeros(2, (n+2));

    Tz = [[1           0]; ...
          [0 -(phi2+dp2)]];
    Tu = [[0          0]; ...
          [0 (tht2+dt2)]];
    Tz_del = [ [1 0]; ...
               [0 -1*(dp2 + ptb_phi2)] ];
    Tu_del = [ [0 0]; ...
               [0 (dt2 + ptb_tht2)] ];

    outliere = 0;
    outliern = 0;
    outlieru = 0;
    if outlr == 1
        outliere = 5*sde*randn(1,1);
    end
    if outlr == 2
        outliern = 5*sdn*randn(1,1);
    end
    if outlr == 3
        outlieru = 5*sdu*randn(1,1);
    end
    if outlr == 4
        outliere = 5*sde*randn(1,1);
        outlieru = 5*sdu*randn(1,1);
    end
    if outlr == 5
        outliern = 5*sdn*randn(1,1);
        outlieru = 5*sdu*randn(1,1);
    end
    inde = ceil(rand(n));
    indn = ceil(rand(n));
    eps(inde+2) = eps(inde+2) + outliere;
    eta(indn+2) = eps(indn+2) + outliern;
    u(ind+2)    = u(ind+2)    + outlieru;
    for j = 3:(n+2)
         z(j) = -phi1*z(j-1) - phi2*z(j-2) + tht1*u(j-1) + tht2*u(j-2) + eps(j);
         y(j) = z(j) + eta(j);
    end

    clean1 = [0; 0];
    clean2 = [0; 0];

    for j = 3:(n+2)
        [krein_z(:,j), Po_krein] = state_krein(krein_z(:,(j-1)), y(j-1), Po_krein,
         Ad, C, K, Veta, Veps, G1, G2);
        [krein2_z(:,j), Po_krein2] = state_krein_IR(krein2_z(:,(j-1)), y(j-1),
         Po_krein2, Ad, C, K, Veta, Veps, G1, G2, u(j-1), Bd);
        [krein3_z(:,j), Po_krein3] = state_krein_IR(krein3_z(:,(j-1)), y(j-1),
         Po_krein3, Ad, C, K2, Veta, Veps, G1_2, G2, u(j-1), Bd);
        [kalman_z(:,j), Po_kalm] = State_Kalman(Ad, Bd, G2, kalman_z(:,(j-1)),
         Po_kalm, u(j-1), C, y(j-1), Veps, Veta);
        index = -(j-1):-(j-2);
        naive_z(:,j) = State_Naive(u(-1*index), y(-1*index), Tu, Tz, Ad, Bd);
        [naiveE_z(:,j), clean1] = State_NaiveE(u(-1*index), y(-1*index), clean1,
         Tu, Tz, Ad, Bd, lambda);
        [naiveEBC_z(:,j), clean2] = State_NaiveEBC(u(-1*index), y(-1*index), clean2,
         Tu, Tz, Ad, Bd, Adelt, Bdelt, Tz_del, Tu_del, lambda);
        naiveBC_z(:,j) = State_NaiveBC(u(-1*index), y(-1*index), Tu, Tz, Ad, Bd,
         Adelt, Bdelt, Tz_del, Tu_del);
    end

    indices = 1:n + 2;
    z = z(indices);
    y = y(indices);
    krein_z = krein_z(:,indices);
    krein2_z = krein2_z(:,indices);
    krein3_z = krein3_z(:,indices);
    kalman_z = kalman_z(:,indices);
    naive_z = naive_z(:,indices);
    naiveBC_z = naiveBC_z(:,indices);
    naiveE_z = naiveE_z(:,indices);
    naiveEBC_z = naiveEBC_z(:,indices);

    krein_ss = sum((krein_z(1,:)' - z).^2) / (n-1);
    krein2_ss = sum((krein2_z(1,:)' - z).^2) / (n-1);
    krein3_ss = sum((krein3_z(1,:)' - z).^2) / (n-1);
    kalm_ss  = sum((kalman_z(1,:)' - z).^2) / (n-1);
    naive_ss = sum((naive_z(1,:)' - z).^2) / (n-1);
    naiveBC_ss = sum((naiveBC_z(1,:)' - z).^2) / (n-1);
    naiveE_ss = sum((naiveE_z(1,:)' - z).^2) / (n-1);
    naiveEBC_ss = sum((naiveEBC_z(1,:)' - z).^2) / (n-1);

    mse = [krein_ss; krein2_ss; krein3_ss; kalm_ss; naive_ss;
     naiveBC_ss; naiveE_ss; naiveEBC_ss];

return;
```

B.2 R Code

```
filenames52 <- scan('C:/My Files/Thesis/Filenames/FilenamesMSE52noP2.txt',
what=character())
len <- length(filenames52)
delta <- c('-0.5', '0', '+0.5')
Delta <- matrix(NA, nrow=len, ncol=4)
L <- 0
for(i in 1:3)
{for(j in 1:3)
{for(k in 1:3)
{for(m in 1:3)
{
L <- L + 1
Delta[L,] <- c(i,j,k,m)
}}}}
filenameplot <- scan('C:/My Files/Thesis/Filenames/FilenamesPlots52two.txt',
what=character())
for(i in 1:len)
{
trellis.par.set( box.umbrella = list(lty = 1) )
A = as.numeric(delta[Delta[i,1]])
B = as.numeric(delta[Delta[i,2]])
C = as.numeric(delta[Delta[i,3]])
D = as.numeric(delta[Delta[i,4]])
data <- scan(filenames52[i], quiet=TRUE)
data <- matrix(data, ncol=8)
grp <- c('K1','K2','K3','K4','N','NBC','NE','NEBC')
grp <- as.factor(rep(grp, each=100))
# png(file=filenameplot[i], width=600, height=480)
print(
bwplot(c(data)~grp, ylab='MSE', pch=19,
par.settings = list(plot.symbol = list(pch = "+")),
main=eval(substitute(expression(paste('        ', delta[phi[1]],
" = ", tval, '  |  ',
delta[phi[2]], ' = ', tval2, '  |  ',
delta[theta[1]], ' = ', tval3, '  |  ',
delta[theta[2]], ' = ', tval4,)),
list(tval=A, tval2=B, tval3=C, tval4=D))))
)
dev.print(file=filenameplot[i], device=png, width=600, height=480)
dev.off()
}

filenames12 <- scan('C:/My Files/Thesis/Filenames/FilenamesMSE12noP2.txt',
what=character())
len <- length(filenames12)
delta <- c('-0.5', '0', '+0.5')
Delta <- matrix(NA, nrow=len, ncol=4)
L <- 0
for(i in 1:3)
{for(j in 1:3)
{for(k in 1:3)
{for(m in 1:3)
{
L <- L + 1
Delta[L,] <- c(i,j,k,m)
}}}}
filenameplot <- scan('C:/My Files/Thesis/Filenames/FilenamesPlots12two.txt',
what=character())
for(i in 1:len)
{
trellis.par.set( box.umbrella = list(lty = 1) )
A = as.numeric(delta[Delta[i,1]])
B = as.numeric(delta[Delta[i,2]])
C = as.numeric(delta[Delta[i,3]])
D = as.numeric(delta[Delta[i,4]])
data <- scan(filenames12[i], quiet=TRUE)
data <- matrix(data, ncol=8)
grp <- c('K1','K2','K3','K4','N','NBC','NE','NEBC')
grp <- as.factor(rep(grp, each=100))
# png(file=filenameplot[i], width=600, height=480)
print(
bwplot(c(data)~grp, ylab='MSE', pch=19,
par.settings = list(plot.symbol = list(pch = "+")),
main=eval(substitute(expression(paste('        ', delta[phi[1]],
" = ", tval, '  |  ',
delta[phi[2]], ' = ', tval2, '  |  ',
delta[theta[1]], ' = ', tval3, '  |  ',
delta[theta[2]], ' = ', tval4,)),
list(tval=A, tval2=B, tval3=C, tval4=D))))
)
dev.print(file=filenameplot[i], device=png, width=600, height=480)
dev.off()
}



# Compile data for GLM usage.

comb <- matrix(NA, nrow=(81*8), ncol=5)
L <- 0
for(i in 1:3)
{for(j in 1:3)
{for(k in 1:3)
{for(l in 1:3)
{for(m in 1:8)
{
L <- L + 1
comb[L,] <- c(i, j, k, l, m)
}}}}}


filenames52 <- scan('C:/My Files/Thesis/Filenames/FilenamesMSE52noP2.txt',
what=character())
len <- length(filenames52)
data52 <- matrix(NA, nrow=len*100*8, ncol=6)
ind <- list()
for(i in 1:len)
{
print(i)
mx <- 100*8*i
ind1 <- 8*i
mn <- mx - 100*8
ind2 <- ind1-8
ind[[i]] <- (mn+1):mx

dat <- scan(filenames52[i], quiet=TRUE)
dat <- c(matrix(dat, ncol=8))
Comb <- matrix(rep(comb[(ind2+1):ind1,], each=100), nrow=8*100)
data52[ind[[i]],1] <- dat
data52[ind[[i]],2:6] <- Comb
}

filenames12 <- scan('C:/My Files/Thesis/Filenames/FilenamesMSE12noP2.txt',
what=character())
len <- length(filenames12)
data12 <- matrix(NA, nrow=len*100*8, ncol=6)
ind <- list()
for(i in 1:len)
{
print(i)
mx <- 100*8*i
ind1 <- 8*i
mn <- mx - 100*8
ind2 <- ind1-8
```

```
ind[[i]] <- (mn+1):mx

dat <- scan(filenames12[i], quiet=TRUE)
dat <- c(matrix(dat, ncol=8))
Comb <- matrix(rep(comb[(ind2+1):ind1,], each=100), nrow=8*100)
data12[ind[[i]],1] <- dat
data12[ind[[i]],2:6] <- Comb
}
NR <- nrow(data52)
Data <- cbind(rbind(data12, data52), rep(c(1,2), each=NR))

# The following function creates the essence matrix for use with converting
# from the reference cell coding to cell means.  It can really be used for
# other reasons as well, pre-multiplying the essence matrix from model.matrix
# by the coefficients will produce cell means for any coding scheme for model
# fit.  It is also useful for just viewing the X or design matrix to see what
# the fit is doing and should be compatible with most linear fit functions in R.

essence.X <- function(fit)
{
X <- model.matrix(fit)
nc <- ncol(X)
X <- unique(apply(X, 1, function(x) paste(x, collapse=':')))
matrix(as.numeric(unlist(strsplit(X, split=':'))), ncol=nc, byrow=TRUE)
}

y <- Data[,1]
pd1 <- as.factor(Data[,2])
pd2 <- as.factor(Data[,3])
td1 <- as.factor(Data[,4])
td2 <- as.factor(Data[,5])
flt <- as.factor(Data[,6])
ssz <- as.factor(Data[,7])

# Fit a Generalized Linear Model with main effects and first order interactions.
#options(contrasts=c('contr.treatment','contr.sum')) # Doesn't work, (too little memory to do effects in model).

# The R default contr.treatment, which is used here for 'ordered', since we have ordered data by factor in
# the model.  This default uses the first level of factors as the reference cell.
gc()
options(contrasts=c('contr.treatment','contr.treatment'))
fit <- lm(y ~ pd1 + pd2 + td1 + td2 + flt + ssz +
pd1:pd2 + pd1:td1 + pd1:td2 + pd1:flt + pd1:ssz +
pd2:td1 + pd2:td2 + pd2:flt + pd2:ssz +
td1:td2 + td1:flt + td2:ssz +
td2:flt + td2:ssz +
flt:ssz)

gc()

# parse the data and do a biglm
library(biglm)
options(contrasts=c('contr.sum','contr.sum'))
#options(contrasts=c('contr.treatment','contr.treatment'))
indx <- seq(1, 129600, by=100)
data2 <- Data[indx,]
y <- data2[,1]
a <- as.factor(data2[,2])
b <- as.factor(data2[,3])
c <- as.factor(data2[,4])
d <- as.factor(data2[,5])
e <- as.factor(data2[,6])
f <- as.factor(data2[,7])
Data2 <- as.data.frame(cbind(y, a, b, c, d, e, f))
Data2$a <- as.factor(Data2$a)
Data2$b <- as.factor(Data2$b)
Data2$c <- as.factor(Data2$c)
Data2$d <- as.factor(Data2$d)
```

```
Data2$e <- as.factor(Data2$e)
Data2$f <- as.factor(Data2$f)


ff <- y ~ a + b + c + d + e + f +
a:b + a:c + a:d + a:e + a:f +
b:c + b:d + b:e + b:f +
c:d + c:e + c:f +
d:e + d:f +
e:f
bigfit <- biglm(ff,Data2)

for(i in 2:99)
{
print(i)
indx <- seq(1, 129600, by=100) + i
data2 <- Data[indx,]
y <- data2[,1]
a <- as.factor(data2[,2])
b <- as.factor(data2[,3])
c <- as.factor(data2[,4])
d <- as.factor(data2[,5])
e <- as.factor(data2[,6])
f <- as.factor(data2[,7])
Data2 <- as.data.frame(cbind(y, a, b, c, d, e, f))
Data2$a <- as.factor(Data2$a)
Data2$b <- as.factor(Data2$b)
Data2$c <- as.factor(Data2$c)
Data2$d <- as.factor(Data2$d)
Data2$e <- as.factor(Data2$e)
Data2$f <- as.factor(Data2$f)

bigfit <- update(bigfit,Data2)
}

options(contrasts=c('contr.sum','contr.sum'))
#options(contrasts=c('contr.treatment','contr.treatment'))
indx <- seq(1, 129600, by=100)
data2 <- Data[indx,]
y <- data2[,1]
a <- as.factor(data2[,2])
b <- as.factor(data2[,3])
c <- as.factor(data2[,4])
d <- as.factor(data2[,5])
e <- as.factor(data2[,6])
f <- as.factor(data2[,7])
Data2 <- as.data.frame(cbind(y, a, b, c, d, e, f))
Data2$a <- as.factor(Data2$a)
Data2$b <- as.factor(Data2$b)
Data2$c <- as.factor(Data2$c)
Data2$d <- as.factor(Data2$d)
Data2$e <- as.factor(Data2$e)
Data2$f <- as.factor(Data2$f)


ff2 <- y ~ a:b:c:d:e:f - 1
bigfitcm <- biglm(ff2,Data2)

for(i in 2:99)
{
print(i)
indx <- seq(1, 129600, by=100) + i
data2 <- Data[indx,]
y <- data2[,1]
a <- as.factor(data2[,2])
b <- as.factor(data2[,3])
c <- as.factor(data2[,4])
d <- as.factor(data2[,5])
```

```
e <- as.factor(data2[,6])
f <- as.factor(data2[,7])
Data2 <- as.data.frame(cbind(y, a, b, c, d, e, f))
Data2$a <- as.factor(Data2$a)
Data2$b <- as.factor(Data2$b)
Data2$c <- as.factor(Data2$c)
Data2$d <- as.factor(Data2$d)
Data2$e <- as.factor(Data2$e)
Data2$f <- as.factor(Data2$f)

bigfitcm <- update(bigfitcm,Data2)
}

smry <- summary(fit)
infer <- smry$fstatistic
pval <- 1 - pf(infer[1], infer[2], infer[3])
# This anova() for reference cell should produce the exact same output
# as the effects coding anova.
anvfit <- anova(fit)
smryeffects <- summary(bigfit)
smrycm <- summary(bigfitcm)
datinfer <- smryeffects$mat
pltdat <- datinfer[,1:3]
pval <- datinfer[,5]

mx <- max(pltdat[,3])
mn <- min(pltdat[,2])
nrp <- nrow(pltdat)
plot(pltdat[2,2], nrp, pch='(', col='blue', xlim=c(mn, mx), cex=1,
ylim=c(0, (nrp+20)), type='n', axes=FALSE, ylab='',
xlab='Parameter Effect On MSE', main='95% Confidence Intervals')
alph <- 0.05
for(i in 2:nrp)
{
if(i <= 17 & pval[i] < alph)
{
Col <- 'blue'
cex1 <- 0.75
}
if(i <= 17 & pval[i] >= alph)
{
Col <- 'red'
cex1 <- 1.25
}
if(i > 17 & pval[i] < alph)
{
Col <- 'purple'
cex1 <- 0.75
}
if(i > 17 & pval[i] >= alph)
{
Col <- 'red'
cex1 <- 1.25
}
points(pltdat[i,2], nrp-i, pch='(', col=Col, cex=cex1)
points(pltdat[i,3], nrp-i, pch=')', col=Col, cex=cex1)
}

abline(h=nrp-17.5, lty=2)
mtext(side=2, at=(nrp+20-15), text='Main Effects', col='blue', line=1)
mtext(side=2, at=(nrp-17.5)/2, text='2-Way Interactions',
col='purple', line=1)
axis(1, pretty(c(mn, mx)), pretty(c(mn, mx)))
abline(v=0, lty=3)
box()
legend('bottomright', c(expression(paste(H[o], '  Both')),
expression(paste(H[a], '  Main Effect')),
expression(paste(H[a], '  Interaction'))), col=c('red','blue','purple'),
pch=c(19,19,19), bty='n')

cfcm <- as.numeric(smrycm$mat[,1])
matflte <- diag(1,8)
matflt1 <- c(1,-1,0,0,0,0,0,0)
matflt2 <- c(1,0,-1,0,0,0,0,0)
matflt3 <- c(1,0,0,-1,0,0,0,0)
matflt4 <- c(1,0,0,0,-1,0,0,0)
matflt5 <- c(1,0,0,0,0,-1,0,0)
matflt6 <- c(1,0,0,0,0,0,-1,0)
matflt7 <- c(1,0,0,0,0,0,0,-1)
matflt  <- rbind(matflt1,matflt2,matflt3,matflt4,matflt5,matflt6,matflt7)
contr.flt <- t(c(1,1)) %x% matflt %x% t(rep(1, 81))/(3^4*2)
contr.flte <- t(c(1,1)) %x% matflte %x% t(rep(1, 81))/(3^4*2)
X <- model.matrix(bigfitcm)
estfltcont <- contr.flt %*% as.matrix(cfcm)
estflt <- contr.flte %*% as.matrix(cfcm)

contr.ssz <- c(1,-1) %x% rep(1,3^4*8)/(3^4*8)

inter.ssz <- matrix(NA, 7, 1296)
n <- 129600
p <- 3^4*8*2
dendf <- (p*(n-1))
sigsq <- smrycm$obj$qr$ss/dendf

for(i in 1:7)
{
inter.ssz[i,] <- t( as.matrix(contr.flt[i,]) * as.matrix(contr.ssz) )
}

tt <- numeric(7)
ptt <- numeric(7)
rr <- qr(inter.ssz)$rank
CC.int <- inter.ssz
est.int <- CC.int%*%cfcm
F.int <- (t(CC.int%*%cfcm)%*%solve(CC.int%*%solve(t(X)%*%X)%*%t(CC.int))%*%CC.int%*%cfcm/rr)/sigsq
pv.int <- 1-pf(F.int, rr, dendf)
low.int <- numeric(7)
up.int <- numeric(7)
Fstat.int <- qf(1-0.05, rr, dendf)
for(i in 1:7)
{
tval <- qt(1-0.05/2, dendf)
low.int[i] <- est.int[i] - sqrt(sigsq*t(c(CC.int[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.int[i,]))*tval
up.int[i] <- est.int[i] + sqrt(sigsq*t(c(CC.int[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.int[i,]))*tval
tt[i] <- est.int[i] / sqrt(sigsq*t(c(CC.int[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.int[i,]))
ptt[i] <- 1-pt(abs(tt[i]), dendf)
}

cc113 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^3*8*2)%x%c(1,0,0)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc123 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^3*8*2)%x%c(0,1,0)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc133 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^3*8*2)%x%c(0,0,1)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc213 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^2*8*2)%x%c(1,0,0)%x%rep(1,3)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc223 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^2*8*2)%x%c(0,1,0)%x%rep(1,3)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc233 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^2*8*2)%x%c(0,0,1)%x%rep(1,3)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc313 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3*8*2)%x%c(1,0,0)%x%rep(1,3^2)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc323 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3*8*2)%x%c(0,1,0)%x%rep(1,3^2)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc333 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3*8*2)%x%c(0,0,1)%x%rep(1,3^2)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
```

```
cc413 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 8*2)%x%c(1,0,0)%x%rep(1,3^3)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc423 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 8*2)%x%c(0,1,0)%x%rep(1,3^3)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc433 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 8*2)%x%c(0,0,1)%x%rep(1,3^3)/(3^3*8*2) -
c(1,1)%x%c(0,0,1,0,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))

CC1.3 <- rbind(cc113,cc123,cc133)
CC2.3 <- rbind(cc213,cc223,cc233)
CC3.3 <- rbind(cc313,cc323,cc333)
CC4.3 <- rbind(cc413,cc423,cc433)
int1.3 <- CC1.3%*%cfcm
int2.3 <- CC2.3%*%cfcm
int3.3 <- CC3.3%*%cfcm
int4.3 <- CC4.3%*%cfcm

tt.3 <- numeric(12)
ptt.3 <- numeric(12)
rr1.3 <- qr(CC1.3)$rank
F1.3 <- (t(CC1.3%*%cfcm)%*%solve(CC1.3%*%solve(t(X)%*%X)%*%t(CC1.3))%*%CC1.3%*%cfcm/rr1.3)/sigsq
pv1.3 <- 1-pf(F1.3, rr1.3, dendf)
rr2.3 <- qr(CC2.3)$rank
F2.3 <- (t(CC2.3%*%cfcm)%*%solve(CC2.3%*%solve(t(X)%*%X)%*%t(CC2.3))%*%CC2.3%*%cfcm/rr2.3)/sigsq
pv2.3 <- 1-pf(F2.3, rr2.3, dendf)
rr3.3 <- qr(CC3.3)$rank
F3.3 <- (t(CC3.3%*%cfcm)%*%solve(CC3.3%*%solve(t(X)%*%X)%*%t(CC3.3))%*%CC3.3%*%cfcm/rr3.3)/sigsq
pv3.3 <- 1-pf(F3.3, rr3.3, dendf)
rr4.3 <- qr(CC4.3)$rank
F4.3 <- (t(CC4.3%*%cfcm)%*%solve(CC4.3%*%solve(t(X)%*%X)%*%t(CC4.3))%*%CC4.3%*%cfcm/rr4.3)/sigsq
pv4.3 <- 1-pf(F4.3, rr4.3, dendf)

low.int.3 <- numeric(12)
up.int.3 <- numeric(12)
CC.3 <- rbind(CC1.3,CC2.3,CC3.3,CC4.3)
int.3 <- c(int1.3,int2.3,int3.3,int4.3)
for(i in 1:12)
{
tval <- qt(1-0.05/2, dendf)
low.int.3[i] <- int.3[i] - sqrt(sigsq*t(c(CC.3[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.3[i,]))*tval
up.int.3[i] <- int.3[i] + sqrt(sigsq*t(c(CC.3[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.3[i,]))*tval
tt.3[i] <- int.3[i] / sqrt(sigsq*t(c(CC.3[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.3[i,]))
ptt.3[i] <- 1-pt(abs(tt.3[i]), dendf)
}

cc118 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^3*8*2)%x%c(1,0,0)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc128 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^3*8*2)%x%c(0,1,0)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc138 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^3*8*2)%x%c(0,0,1)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc218 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^2*8*2)%x%c(1,0,0)%x%rep(1,3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc228 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^2*8*2)%x%c(0,1,0)%x%rep(1,3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc238 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^2*8*2)%x%c(0,0,1)%x%rep(1,3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc318 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3*8*2)%x%c(1,0,0)%x%rep(1,3^2)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc328 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3*8*2)%x%c(0,1,0)%x%rep(1,3^2)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc338 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3*8*2)%x%c(0,0,1)%x%rep(1,3^2)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc418 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 8*2)%x%c(1,0,0)%x%rep(1,3^3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc428 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 8*2)%x%c(0,1,0)%x%rep(1,3^3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
cc438 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 8*2)%x%c(0,0,1)%x%rep(1,3^3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,0,0,0,0,1)%x%rep(1, 3^4)/(3^4*2))
```

```
CC1.8 <- rbind(cc118,cc128,cc138)
CC2.8 <- rbind(cc218,cc228,cc238)
CC3.8 <- rbind(cc318,cc328,cc338)
CC4.8 <- rbind(cc418,cc428,cc438)
int1.8 <- CC1.8%*%cfcm
int2.8 <- CC2.8%*%cfcm
int3.8 <- CC3.8%*%cfcm
int4.8 <- CC4.8%*%cfcm

tt.8 <- numeric(12)
ptt.8 <- numeric(12)
rr1.8 <- qr(CC1.8)$rank
F1.8 <- (t(CC1.8%*%cfcm)%*%solve(CC1.8%*%solve(t(X)%*%X)%*%t(CC1.8))%*%CC1.8%*%cfcm/rr1.8)/sigsq
pv1.8 <- 1-pf(F1.8, rr1.8, dendf)
rr2.8 <- qr(CC2.8)$rank
F2.8 <- (t(CC2.8%*%cfcm)%*%solve(CC2.8%*%solve(t(X)%*%X)%*%t(CC2.8))%*%CC2.8%*%cfcm/rr2.8)/sigsq
pv2.8 <- 1-pf(F2.8, rr2.8, dendf)
rr3.8 <- qr(CC3.8)$rank
F3.8 <- (t(CC3.8%*%cfcm)%*%solve(CC3.8%*%solve(t(X)%*%X)%*%t(CC3.8))%*%CC3.8%*%cfcm/rr3.8)/sigsq
pv3.8 <- 1-pf(F3.8, rr3.8, dendf)
rr4.8 <- qr(CC4.8)$rank
F4.8 <- (t(CC4.8%*%cfcm)%*%solve(CC4.8%*%solve(t(X)%*%X)%*%t(CC4.8))%*%CC4.8%*%cfcm/rr4.8)/sigsq
pv4.8 <- 1-pf(F4.8, rr4.8, dendf)

low.int.8 <- numeric(12)
up.int.8 <- numeric(12)
CC.8 <- rbind(CC1.8,CC2.8,CC3.8,CC4.8)
int.8 <- c(int1.8,int2.8,int3.8,int4.8)
for(i in 1:12)
{
tval <- qt(1-0.05/2, dendf)
low.int.8[i] <- int.8[i] - sqrt(sigsq*t(c(CC.8[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.8[i,]))*tval
up.int.8[i] <- int.8[i] + sqrt(sigsq*t(c(CC.8[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.8[i,]))*tval
tt.8[i] <- int.8[i] / sqrt(sigsq*t(c(CC.8[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.8[i,]))
ptt.8[i] <- 1-pt(abs(tt.8[i]), dendf)
}

cc114 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^3*8*2)%x%c(1,0,0)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc124 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^3*8*2)%x%c(0,1,0)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc134 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^3*8*2)%x%c(0,0,1)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc214 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^2*8*2)%x%c(1,0,0)%x%rep(1,3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc224 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^2*8*2)%x%c(0,1,0)%x%rep(1,3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc234 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3^2*8*2)%x%c(0,0,1)%x%rep(1,3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc314 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3*8*2)%x%c(1,0,0)%x%rep(1,3^2)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc324 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3*8*2)%x%c(0,1,0)%x%rep(1,3^2)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc334 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 3*8*2)%x%c(0,0,1)%x%rep(1,3^2)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc414 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 8*2)%x%c(1,0,0)%x%rep(1,3^3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc424 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 8*2)%x%c(0,1,0)%x%rep(1,3^3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))
cc434 <- (rep(1, 1296)/1296 - contr.flte[3,]) - (rep(1, 8*2)%x%c(0,0,1)%x%rep(1,3^3)/(3^3*8*2) -
c(1,1)%x%c(0,0,0,1,0,0,0,0)%x%rep(1, 3^4)/(3^4*2))

CC1.4 <- rbind(cc114,cc124,cc134)
CC2.4 <- rbind(cc214,cc224,cc234)
CC3.4 <- rbind(cc314,cc324,cc334)
CC4.4 <- rbind(cc414,cc424,cc434)
int1.4 <- CC1.4%*%cfcm
```

```
int2.4 <- CC2.4%*%cfcm
int3.4 <- CC3.4%*%cfcm
int4.4 <- CC4.4%*%cfcm

tt.4 <- numeric(12)
ptt.4 <- numeric(12)
rr1.4 <- qr(CC1.4)$rank
F1.4 <- (t(CC1.4%*%cfcm)%*%solve(CC1.4%*%solve(t(X)%*%X)%*%t(CC1.4))%*%CC1.4%*%cfcm/rr1.4)/sigsq
pv1.4 <- 1-pf(F1.4, rr1.4, dendf)
rr2.4 <- qr(CC2.4)$rank
F2.4 <- (t(CC2.4%*%cfcm)%*%solve(CC2.4%*%solve(t(X)%*%X)%*%t(CC2.4))%*%CC2.4%*%cfcm/rr2.4)/sigsq
pv2.4 <- 1-pf(F2.4, rr2.4, dendf)
rr3.4 <- qr(CC3.4)$rank
F3.4 <- (t(CC3.4%*%cfcm)%*%solve(CC3.4%*%solve(t(X)%*%X)%*%t(CC3.4))%*%CC3.4%*%cfcm/rr3.4)/sigsq
pv3.4 <- 1-pf(F3.4, rr3.4, dendf)
rr4.4 <- qr(CC4.4)$rank
F4.4 <- (t(CC4.4%*%cfcm)%*%solve(CC4.4%*%solve(t(X)%*%X)%*%t(CC4.4))%*%CC4.4%*%cfcm/rr4.4)/sigsq
pv4.4 <- 1-pf(F4.4, rr4.4, dendf)

low.int.4 <- numeric(12)
up.int.4 <- numeric(12)
CC.4 <- rbind(CC1.4,CC2.4,CC3.4,CC4.4)
int.4 <- c(int1.4,int2.4,int3.4,int4.4)
for(i in 1:12)
{
tval <- qt(1-0.05/2, dendf)
low.int.4[i] <- int.4[i] - sqrt(sigsq*t(c(CC.4[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.4[i,]))*tval
up.int.4[i] <- int.4[i] + sqrt(sigsq*t(c(CC.4[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.4[i,]))*tval
tt.4[i] <- int.4[i] / sqrt(sigsq*t(c(CC.4[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.4[i,]))
ptt.4[i] <- 1-pt(abs(tt.4[i]), dendf)
}


contr.flt12 <- t(c(1,0)) %x% matflt %x% t(rep(1, 81))/(3^4)
contr.flte12 <- t(c(1,0)) %x% matflte %x% t(rep(1, 81))/(3^4)
contr.flt52 <- t(c(0,1)) %x% matflt %x% t(rep(1, 81))/(3^4)
contr.flte52 <- t(c(0,1)) %x% matflte %x% t(rep(1, 81))/(3^4)
estfltcont12 <- contr.flt12 %*% as.matrix(cfcm)
estflt12 <- contr.flte12 %*% as.matrix(cfcm)
estfltcont52 <- contr.flt52 %*% as.matrix(cfcm)
estflt52 <- contr.flte52 %*% as.matrix(cfcm)
m <- qr(matflt)$rank
me <- qr(matflte)$rank
n <- 129600
p <- 3^4*8*2
dendf <- (p*(n-1))
sigsq <- smrycm$obj$qr$ss/dendf

# General test of contrasts
CC <- contr.flt
F <- (t(CC%*%cfcm)%*%solve(CC%*%solve(t(X)%*%X)%*%t(CC))%*%CC%*%cfcm/m)/sigsq
pv <- 1-pf(F, m, dendf)

CCe <- contr.flte
Fe <- (t(CCe%*%cfcm)%*%solve(CCe%*%solve(t(X)%*%X)%*%t(CCe))%*%CCe%*%cfcm/me)/sigsq
pve <- 1-pf(Fe, me, dendf)

CC12 <- contr.flt12
CCe12 <- contr.flte12
CC52 <- contr.flt52
CCe52 <- contr.flte52
up <- numeric(8)
low <- numeric(8)
tt <- numeric(8)
ptt <- numeric(8)
for(i in 1:8)
{
tval <- qt(1-0.05/2, dendf)
```

```
low[i] <- estflt[i] - sqrt(sigsq*t(c(CCe[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe[i,]))*tval
up[i] <- estflt[i] + sqrt(sigsq*t(c(CCe[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe[i,]))*tval
tt[i] <- estflt[i] / sqrt(sigsq*t(c(CCe[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe[i,]))
ptt[i] <- 1-pt(abs(tt[i]), dendf)
}

mx <- max(up)
mn <- min(low)
nrp <- nrow(estflt)
plot(estflt[1,1], nrp, pch='(', col='blue', xlim=c(0, mx), cex=1,
ylim=c(0, (nrp)), type='n', axes=FALSE, ylab='',
xlab='Filter MSE')
points(low, nrp:1, pch='(', col='blue', cex=1)
points(estflt, nrp:1, pch='+', col='red', cex=0.75)
points(up, nrp:1, pch=')', col='blue', cex=1)
axis(side=2, nrp:1, c('K1','K2','K3','K4','N','NBC','NE','NEBC'), las=2)
axis(1, pretty(c(0, mx), n=8), pretty(c(0, mx), n=8))
abline(v=pretty(c(0, mx), n=8), lty=3, col='lightgrey')
box()

mx <- max(up[c(2,3,7,8)])
mn <- min(low[c(2,3,7,8)])
plot(estflt[1,1], (nrp-4), pch='(', col='blue', xlim=c(mn, mx), cex=1,
ylim=c(0, (nrp-4+1)), type='n', axes=FALSE, ylab='',
xlab='Filter MSE')
points(low[c(2,3,7,8)], (nrp-4):1, pch='(', col='blue', cex=1.25)
points(estflt[c(2,3,7,8)], (nrp-4):1, pch='+', col='red', cex=1.25)
points(up[c(2,3,7,8)], (nrp-4):1, pch=')', col='blue', cex=1.25)
axis(side=2, (nrp-4):1, c('K2','K3','NE','NEBC'), las=2)
axis(1, pretty(c(mn, mx), n=6), pretty(c(mn, mx), n=6))
abline(v=pretty(c(mn, mx), n=8), lty=3, col='lightgrey')
box()

CC2v7 <- contr.flt[1,] - contr.flt[6,]
CC7v8 <- contr.flt[6,] - contr.flt[7,]
CC3v8 <- contr.flt[2,] - contr.flt[7,]
CC28  <- rbind(CC2v7,CC7v8,CC3v8)

Fe2 <- (t(CC28%*%cfcm)%*%solve(CC28%*%solve(t(X)%*%X)%*%t(CC28))%*%CC28%*%cfcm/qr(CC28)$rank)/sigsq
r <- qr(CC28)$rank
pve2 <- 1-pf(Fe2, r, dendf)
tte2 <- numeric(3)
TTe2 <- numeric(3)
ptte2 <- numeric(3)
for(i in 1:3)
{
tte2[i] <- t(c(CC28[i,]))%*%cfcm / sqrt(sigsq*t(c(CC28[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC28[i,]))
ptte2[i] <- 2*(1-pt(abs(tte2[i]), dendf))
TTe2[i] <- qt(1-0.05/(2*r), dendf)
}


up12 <- numeric(8)
low12 <- numeric(8)
tt12 <- numeric(8)
ptt12 <- numeric(8)
for(i in 1:8)
{
tval <- qt(1-0.05/2, dendf)
low12[i] <- estflt12[i] - sqrt(sigsq*t(c(CCe12[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe12[i,]))*tval
up12[i] <- estflt12[i] + sqrt(sigsq*t(c(CCe12[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe12[i,]))*tval
tt12[i] <- estflt12[i] / sqrt(sigsq*t(c(CCe12[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe12[i,]))
ptt12[i] <- 1-pt(abs(tt12[i]), dendf)
}

mx <- max(up12)
mn <- min(low12)
nrp <- nrow(estflt12)
```

```
plot(estflt12[1,1], nrp, pch='(', col='blue', xlim=c(0, mx), cex=1,
ylim=c(0, (nrp)), type='n', axes=FALSE, ylab='',
xlab='Filter MSE')
points(low12, nrp:1, pch='(', col='blue', cex=1.25)
points(estflt12, nrp:1, pch='+', col='red', cex=1)
points(up12, nrp:1, pch=')', col='blue', cex=1.25)
for(i in 1:nrp) lines(cbind(estflt,estflt12)[i,], cbind(nrp:1,nrp:1)[i,], lty=2)
axis(side=2, nrp:1, c('K1','K2','K3','K4','N','NBC','NE','NEBC'), las=2)
axis(1, pretty(c(0, mx), n=8), pretty(c(0, mx), n=8))
abline(v=pretty(c(0, mx), n=8), lty=3, col='lightgrey')
box()

mx <- max(up12[c(2,3,7,8)])
mn <- min(low[c(2,3,7,8)])
plot(estflt12[1,1], (nrp-4), pch='(', col='blue', xlim=c(mn, mx), cex=1,
ylim=c(0, (nrp-4+1)), type='n', axes=FALSE, ylab='',
xlab='Filter MSE')
points(low12[c(2,3,7,8)], (nrp-4):1, pch='(', col='blue', cex=1.25)
points(estflt12[c(2,3,7,8)], (nrp-4):1, pch='+', col='red', cex=1.25)
points(up12[c(2,3,7,8)], (nrp-4):1, pch=')', col='blue', cex=1.25)
for(i in 1:(nrp-4)) lines(cbind(estflt[c(2,3,7,8)],estflt12[c(2,3,7,8)])[i,],
cbind((nrp-4):1,(nrp-4):1)[i,], lty=2)
axis(side=2, (nrp-4):1, c('K2','K3','NE','NEBC'), las=2)
axis(1, pretty(c(mn, mx), n=6), pretty(c(mn, mx), n=6))
abline(v=pretty(c(mn, mx), n=8), lty=3, col='lightgrey')
box()

CC2v712 <- contr.flt12[1,] - contr.flt12[6,]
CC7v812 <- contr.flt12[6,] - contr.flt12[7,]
CC3v812 <- contr.flt12[2,] - contr.flt12[7,]
CC2812  <- rbind(CC2v712,CC7v812,CC3v812)

Fe212 <- (t(CC2812%*%cfcm)%*%solve(CC2812%*%solve(t(X)%*%X)%*%t(CC2812))%*%CC2812%*%cfcm/qr(CC2812)$rank)/sigsq
r <- qr(CC2812)$rank
pve212 <- 1-pf(Fe212, r, dendf)
tte212 <- numeric(3)
TTe212 <- numeric(3)
ptte212 <- numeric(3)
for(i in 1:3)
{
tte212[i] <- t(c(CC2812[i,]))%*%cfcm / sqrt(sigsq*t(c(CC2812[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC2812[i,]))
ptte212[i] <- 2*(1-pt(abs(tte212[i]), dendf))
TTe212[i] <- qt(1-0.05/(2*r), dendf)
}

up52 <- numeric(8)
low52 <- numeric(8)
tt52 <- numeric(8)
ptt52 <- numeric(8)
for(i in 1:8)
{
tval <- qt(1-0.05/2, dendf)
low52[i] <- estflt52[i] - sqrt(sigsq*t(c(CCe52[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe52[i,]))*tval
up52[i] <- estflt52[i] + sqrt(sigsq*t(c(CCe52[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe52[i,]))*tval
tt52[i] <- estflt52[i] / sqrt(sigsq*t(c(CCe52[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe52[i,]))
ptt52[i] <- 1-pt(abs(tt52[i]), dendf)
}

mx <- max(up52)
mn <- min(low52)
nrp <- nrow(estflt12)
plot(estflt52[1,1], nrp, pch='(', col='blue', xlim=c(0, mx), cex=1,
ylim=c(0, (nrp)), type='n', axes=FALSE, ylab='',
xlab='Filter MSE')
points(low52, nrp:1, pch='(', col='blue', cex=1.25)
points(estflt52, nrp:1, pch='+', col='red', cex=1)
points(up52, nrp:1, pch=')', col='blue', cex=1.25)
for(i in 1:nrp) lines(cbind(estflt,estflt52)[i,], cbind(nrp:1,nrp:1)[i,], lty=2)
```

```
axis(side=2, nrp:1, c('K1','K2','K3','K4','N','NBC','NE','NEBC'), las=2)
axis(1, pretty(c(0, mx), n=8), pretty(c(0, mx), n=8))
abline(v=pretty(c(0, mx), n=8), lty=3, col='lightgrey')
box()

mx <- max(up[c(2,3,7,8)])
mn <- min(low52[c(2,3,7,8)])
plot(estflt52[1,1], (nrp-4), pch='(', col='blue', xlim=c(mn, mx), cex=1,
ylim=c(0, (nrp-4+1)), type='n', axes=FALSE, ylab='',
xlab='Filter MSE')
points(low52[c(2,3,7,8)], (nrp-4):1, pch='(', col='blue', cex=1.25)
points(estflt52[c(2,3,7,8)], (nrp-4):1, pch='+', col='red', cex=1.25)
points(up52[c(2,3,7,8)], (nrp-4):1, pch=')', col='blue', cex=1.25)
for(i in 1:(nrp-4)) lines(cbind(estflt[c(2,3,7,8)],estflt52[c(2,3,7,8)])[i,],
cbind((nrp-4):1,(nrp-4):1)[i,], lty=2)
axis(side=2, (nrp-4):1, c('K2','K3','NE','NEBC'), las=2)
axis(1, pretty(c(mn, mx), n=6), pretty(c(mn, mx), n=6))
abline(v=pretty(c(mn, mx), n=8), lty=3, col='lightgrey')
box()



mx <- max(up12[c(2,3,7,8)])
mn <- min(low52[c(2,3,7,8)])
plot(estflt52[1,1], (nrp-4), pch='(', col='blue', xlim=c(mn, mx), cex=1,
ylim=c(0, (nrp-4+1)), type='n', axes=FALSE, ylab='',
xlab='Filter MSE')
#points(low[c(2,3,7,8)], (nrp-4):1, pch='(', col='blue', cex=1.25)
points(estflt[c(2,3,7,8)], (nrp-4):1, pch='|', col='blue', cex=1.5)
#points(up[c(2,3,7,8)], (nrp-4):1, pch=')', col='black', cex=1.25)
#points(low12[c(2,3,7,8)], (nrp-4):1, pch='(', col='blue', cex=1.25)
points(estflt12[c(2,3,7,8)], (nrp-4):1, pch='+', col='red', cex=2)
#points(up12[c(2,3,7,8)], (nrp-4):1, pch=')', col='blue', cex=1.25)
#points(low52[c(2,3,7,8)], (nrp-4):1, pch='(', col='blue', cex=1.25)
points(estflt52[c(2,3,7,8)], (nrp-4):1, pch='*', col='green', cex=2)
#points(up52[c(2,3,7,8)], (nrp-4):1, pch=')', col='blue', cex=1.25)
for(i in 1:(nrp-4)) lines(cbind(estflt[c(2,3,7,8)],estflt12[c(2,3,7,8)])[i,],
cbind((nrp-4):1,(nrp-4):1)[i,], lty=2, col='black')
for(i in 1:(nrp-4)) lines(cbind(estflt[c(2,3,7,8)],estflt52[c(2,3,7,8)])[i,],
cbind((nrp-4):1,(nrp-4):1)[i,], lty=2, col='black')
axis(side=2, (nrp-4):1, c('K2','K3','NE','NEBC'), las=2)
axis(1, pretty(c(mn, mx), n=6), pretty(c(mn, mx), n=6))
abline(v=pretty(c(mn, mx), n=8), lty=3, col='lightgrey')
legend('topleft', c('Trace of 12','Overall','Trace of 52'), pch=c('+','|','*'),
col=c('red','blue','green'), bty='n')
box()



CC2v752 <- contr.flt52[1,] - contr.flt52[6,]
CC7v852 <- contr.flt52[6,] - contr.flt52[7,]
CC3v852 <- contr.flt52[2,] - contr.flt52[7,]
CC2852  <- rbind(CC2v752,CC7v852,CC3v852)

Fe252 <- (t(CC2852%*%cfcm)%*%solve(CC2852%*%solve(t(X)%*%X)%*%t(CC2852))%*%CC2852%*%cfcm/qr(CC2852)$rank)/sigsq
r <- qr(CC2852)$rank
pve252 <- 1-pf(Fe252, r, dendf)
tte252 <- numeric(3)
TTe252 <- numeric(3)
ptte252 <- numeric(3)
for(i in 1:3)
{
tte252[i] <- t(c(CC2852[i,]))%*%cfcm / sqrt(sigsq*t(c(CC2852[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC2852[i,]))
ptte252[i] <- 2*(1-pt(abs(tte252[i]), dendf))
TTe252[i] <- qt(1-0.05/(2*r), dendf)
}

cc.lam2 <- matrix(c(1, -1, 0, 1, 0, -1), nrow=2, byrow=TRUE)
```

```
CC.lam2 <- t(rep(1, 8*2)) %x% cc.lam2 %x% t(rep(1,3^3))
cce.lam2 <- diag(1, 3)
plam2 <- 3^3 * 8 * 2
CCe.lam2 <- t(rep(1, 8*2)) %x% cce.lam2 %x% t(rep(1,3^3)) / plam2
estlam2 <- CCe.lam2%*%cfcm

lowlam <- numeric(3)
uplam <- numeric(3)

for(i in 1:3)
{
tval <- qt(1-0.05/2, dendf)
lowlam[i] <- estlam2[i] - sqrt(sigsq*t(c(CCe.lam2[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe.lam2[i,]))*tval
uplam[i] <- estlam2[i] + sqrt(sigsq*t(c(CCe.lam2[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CCe.lam2[i,]))*tval
}

MSE <- t(smryeffects$mat[1,])

Fcc.lam2 <- (t(CC.lam2%*%cfcm)%*%solve(CC.lam2%*%solve(t(X)%*%X)%*%t(CC.lam2))%*%CC.lam2%*%cfcm/qr(CC.lam2)$rank)/sigsq
r <- qr(CC.lam2)$rank
pcc <- 1-pf(Fcc.lam2, r, dendf)
ttcc <- numeric(2)
TTcc <- numeric(2)
pttcc <- numeric(2)
for(i in 1:2)
{
ttcc[i] <- t(c(CC.lam2[i,]))%*%cfcm / sqrt(sigsq*t(c(CC.lam2[i,]))%*%solve(t(X)%*%X)%*%as.matrix(CC.lam2[i,]))
pttcc[i] <- 2*(1-pt(abs(ttcc[i]), dendf))
TTcc[i] <- qt(1-0.05/(2*r), dendf)
}

# To validate biglm does the same thing as lm.

options(contrasts=c('contr.sum','contr.sum'))
indx <- seq(1, 129600, by=100)
data2 <- Data[indx,]
y <- data2[,1]
a <- as.factor(data2[,2])
b <- as.factor(data2[,3])
c <- as.factor(data2[,4])
d <- as.factor(data2[,5])
e <- as.factor(data2[,6])
f <- as.factor(data2[,7])
Data2 <- as.data.frame(cbind(y, a, b, c, d, e, f))
Data2$a <- as.factor(Data2$a)
Data2$b <- as.factor(Data2$b)
Data2$c <- as.factor(Data2$c)
Data2$d <- as.factor(Data2$d)
Data2$e <- as.factor(Data2$e)
Data2$f <- as.factor(Data2$f)


ff <- y ~ a + b + a:b
bigfit2 <- biglm(ff,Data2)

for(i in 2:99)
{
indx <- seq(1, 129600, by=100) + i
data2 <- Data[indx,]
y <- data2[,1]
a <- as.factor(data2[,2])
b <- as.factor(data2[,3])
c <- as.factor(data2[,4])
d <- as.factor(data2[,5])
e <- as.factor(data2[,6])
f <- as.factor(data2[,7])
Data2 <- as.data.frame(cbind(y, a, b, c, d, e, f))
Data2$a <- as.factor(Data2$a)
Data2$b <- as.factor(Data2$b)
Data2$c <- as.factor(Data2$c)
Data2$d <- as.factor(Data2$d)
Data2$e <- as.factor(Data2$e)
Data2$f <- as.factor(Data2$f)

bigfit2 <- update(bigfit2,Data2)
}
y <- Data[,1]

fit2 <- lm(y ~ pd1*pd2)
```