Studienarbeit

Multiple Model Adaptive Control of Muscle Relaxation in Anesthesia

Autor:

Anne-Kathrin Hess

gebohren am 08.03.1986

in Erfurt, Germany

Matrikelnummer 172771

Betreuer:

Prof. Guy A. Dumont

University of British Columbia

Prüfer:

Prof. Dr. A. Kienle

Otto-von-Guerick-Universität Magdeburg

Juli 2008

© Anne-Kathrin Hess, 2008.

Contents

1 Introduction **3**

2 Patient Models **5**

2.1 Laguerre Polynomials 6

2.2 Input and Output 7

2.3 Open Loop Behavior 7

3 Hybrid Observer **9**

3.1 Kalman Filter 10

3.2 Multiple Model Adaptive Estimation (MMAE) 11

3.2.1 Calculation of $V_k(n)$ 12

3.2.2 Improvements 14

3.2.3 Algorithms 15

3.3 Simulations 15

4 Switched Controller **19**

4.1 Generalized Predictive Control (GPC) 20

4.2 Controller Tuning 24

4.2.1 Open Loop Analysis 24

4.2.2 Closed Loop Analysis 25

4.2.3 Tuning Algorithms 27

4.2.4 Conclusions for Tuning 28

5 Closed Loop Simulations **32**

6 Conclusion **36**

A Renumbering of Models **37**

B Closed loop data **38**

1 Introduction

Anesthesia is a complicated medical process involving several drugs for different purposes. One important aspect of it is the relaxation of muscles before and during surgery. This is usually done by an anesthesiologist who administers the drug Rocuronium to the patient. Initially, a large bolus is given to saturate the patient causing the muscles to relax quickly. Because different people vary in their reaction to Rocuronium, the return of muscle function is used to identify the injection time for the second bolus. When muscle action is observed, another large dose of the drug is given to the patient, usually causing saturation again (A.D. McLeod [4]). The large amounts of drug involved in this process show that an overdose of Rocuronium is not poisonous to the patient. The problems of this method arise mainly from the long saturation times. One example is the inability to recover the patient from paralysis if needed during surgery. Another issue is the possibly long recovery time after the surgery. Additionally muscle movement becomes very likely during the surgery if this drug procedure is used. This can be very unpleasant for both the patient and the medical staff.

It is believed that closed-loop control can improve this procedure. The use of computer control could decrease the usage of drugs that would lower the cost of surgery and reduce the impact of the drugs to the patient. Additionally, closed-loop control can help maintain an even relaxation level during the surgery, decreasing the time periods in which the patient is saturated (T. J. Gilhuly [3]). Therefore, the goal of this work is to achieve a closed-loop control which provides fast relaxation of the muscles initially and holds a constant level of paralysis throughout surgery. The objective is to minimize the time of drug saturation.

The first step to a closed-loop control is the identification of the system model. The main problem of modeling muscle relaxation is the large variability between different patients and “it is unlikely that this variability can be reduced below a minimum level, [...]” (Gepts [5]) even by using parameters which model specific patient differences. The control of the muscle relaxation during surgery introduces additional problems. Unexpected events in the operating room, such as immense blood loss or respiration problems can influence the dynamics of the system and can lead to time variant parameters or unexpected disturbances. To address all these issues, a hybrid system approach in connection with a switched controller will be used in this work. Therefore, the patient is modeled as a hybrid system with m different modes and arbitrary switches between them. The different modes of this system are linear Laguerre models which represent the muscle relaxation dynamics of different patients (see Fig. 1.1). These models were identified in [3] by using real surgery data. During the closed-loop control, the model that best matches the patient is identified in every time step. This model selection is done by a hybrid observer which calculates the discrete state estimate $\hat{\mu}$, with $\hat{\mu}$ being the best matching model. This estimator is additionally able to calculate the state estimate $\hat{x}_k$ of every model k by using m different Kalman filters. Both information are used by the switched controller to

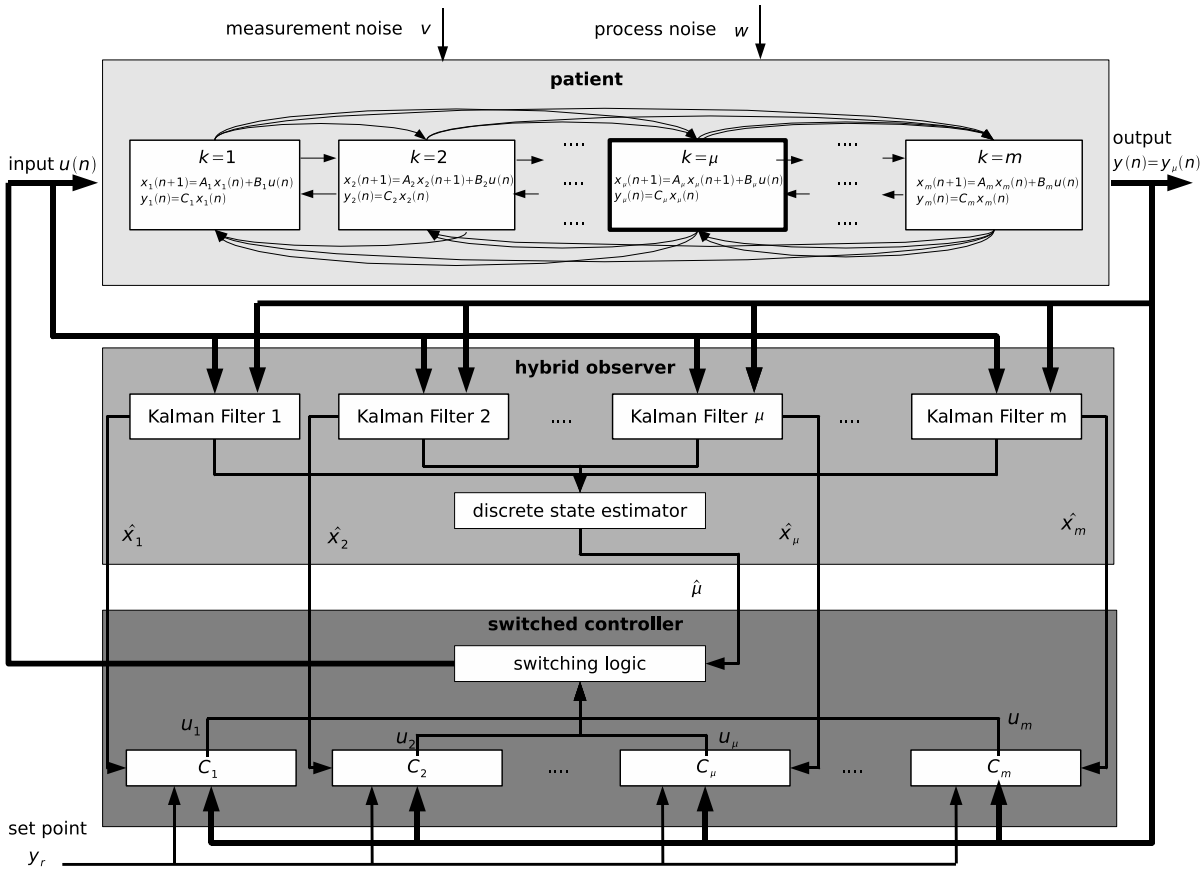

Fig. 1.1: Closed loop (schematic)

calculate the input $u(n)$. This part of the closed loop consists of m separate controllers C_k, each tuned for one model. Based on the discrete state estimate $\hat{\mu}$ the controller tuned for model $\hat{\mu}$ is selected and calculates the input $u(n)$ by using the continuous state estimate $\hat{x}_{\hat{\mu}}$, the output y and the setpoint y_r (see Fig. 1.1).

This algorithm uses a large number of different patient models to handle patient variability. The ability to adapt to unexpected disturbances as well as parameter changes is achieved by allowing switches between the different models during the entire duration of the surgery. Conclusions about the applicability of the proposed algorithm to muscle relaxation in anesthesia will be drawn based on simulation results. First, patient models will be introduced in chapter 2. Afterwards different hybrid observers are introduced in chapter 3 and their performance is compared during simulations. These results are then used with the controller introduced in chapter 4, to simulate the closed-loop performance of the system in chapter 5. Finally, conclusions and suggestions about useful further investigations are presented.

2 Patient Models

It is assumed that the patient can be modeled as a hybrid system with modes that are represented by different linear Laguerre models (see Fig. 2.1). These models were identified in [3] using real surgery data and represent the muscle relaxation dynamics of different monitored patients.

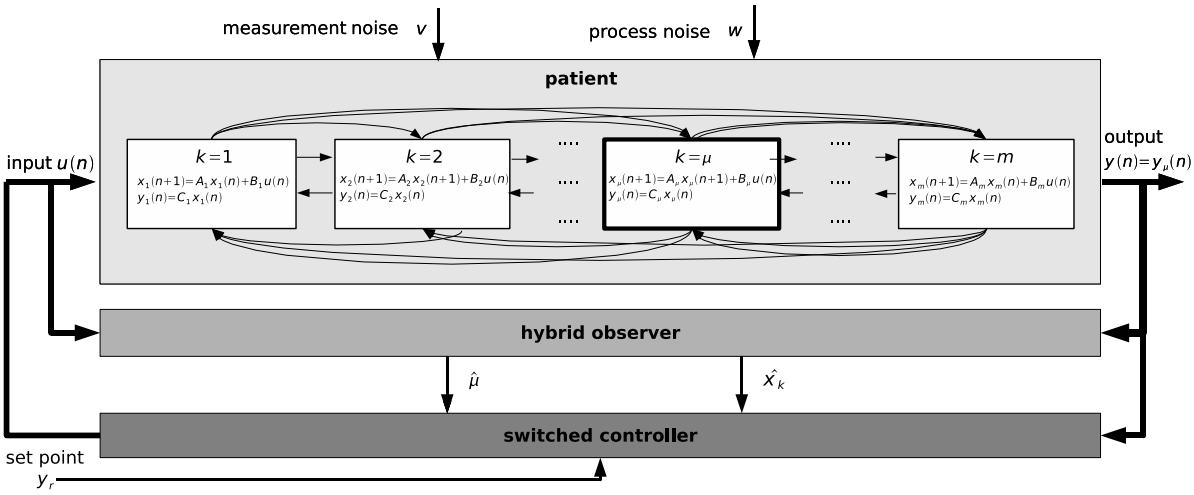

Fig. 2.1: Patient model as a hybrid system (schematic)

Arbitrary switches between the different models are allowed to account for parameter changes or disturbances during surgery. This leads to the following system representation

$$x_k(n+1) = A_k x_k(n) + B_k u(n) \qquad k \in \mathbb{N},\, k = 1 \dots m \tag{2.1}$$
$$y_k(n) = C_k x_k(n)$$

with $m = 59$ being the total number of models used and k being the discrete parameter of the system. As can be seen in (2.1), every mode k has its own system matrices A_k, B_k and C_k and can be interpreted as a discrete-time, time-invariant linear plant. The measured output is assumed to be produced by a true model μ that represents the current patient dynamics.

$$y(n) = y_\mu(n) \tag{2.2}$$

It will be shown in this section how the system matrices of the different models are identified and how the output y is measured. The characteristics of the patient models are discussed with the help of their step responses.

2.1 Laguerre Polynomials

Laguerre models are black box models which do not include any information about the pharmacological dynamics involved in muscle relaxation. Only input and output data collected during surgery were used in [3] to identify the system dynamics. The basic idea of this identification technique is to rebuild the shape of the impulse response $S(t)$ by using a weighted finite sum of orthonormal Laguerre polynomials l_i (similar to a Fourier representation).

$$S(t) = \sum_{i=1}^{N} c_i l_i \tag{2.3}$$

To express these dynamics in form of a linear model (2.4) the Laguerre polynomials l_i are used as states x_i of the system. Since $N = 6$ polynomials are used in [3], the model has 6 states.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2.4}$$

This implies that the weights c_i are combined in the C matrix. Furthermore, the A and B matrices are used to calculate the Laguerre polynomials x_i recursively. The matrices A and B are built by using the Laguerre pole p of the system which is identified along with the weights c_i by using the input and output data of the system.

$$A_{ij} = \begin{cases} p & \text{if } i = j \\ (1-p^2)(-p)^{i-j-i} & \text{if } i > j \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

$$B_i = \sqrt{i - p^2}(-p)^{i-1}$$

$$C_i = \begin{bmatrix} c_1 & c_2 & \dots & c_m \end{bmatrix}$$

One advantage of this method is that it leads to a convenient linear system representation (2.4) even if the real plant includes nonlinearities. Furthermore, filters similar to a Padé approximation are used to calculate the Laguerre polynomials. This is useful for identifying time delays without using them explicitly in the model representation. For more information concerning the use and calculation of Laguerre models please refer to [6] or other related work.

2.2 Input and Output

Based on the input and output data used in [3] to identify the models, the infusion rate of Rocuronium to the patient and the level of muscle relaxation are used as system input u and output y respectively. A normal dose of Rocuronium is $0.6\,mg/kg$ according to the manufacturers recommendation (T.Gilhuly [3]) and therefore the upper bound of the input is set to $u_{max} = 0.8\,mg/kg$. Since the drug can only be injected to and not taken out of the patient, the infusion rate has to be positive. Therefore the lower bound is $u_{min} = 0$.

The value of y is obtained by measuring the response of muscles in the human forearm to stimuli. This measured degree of relaxation is normalized to a value between zero and one. With this scale a relaxation of zero means full muscle function, whereas a relaxation of one means full paralysis of the patient. The desired value of muscle relaxation throughout the surgery is $y = 0.9$ and will be used as a setpoint for the controller. One problem of the output measurement is the muscle recovering time of 20 seconds after each stimuli. This means that an output measurement is only available every 20 seconds. In addition high measurement noise is also problematic. Measurement data published in [3] shows a standard deviation of y to be approximately $\sigma = 0.3$. This value will be used as a parameter for the Kalman Filters.

2.3 Open Loop Behavior

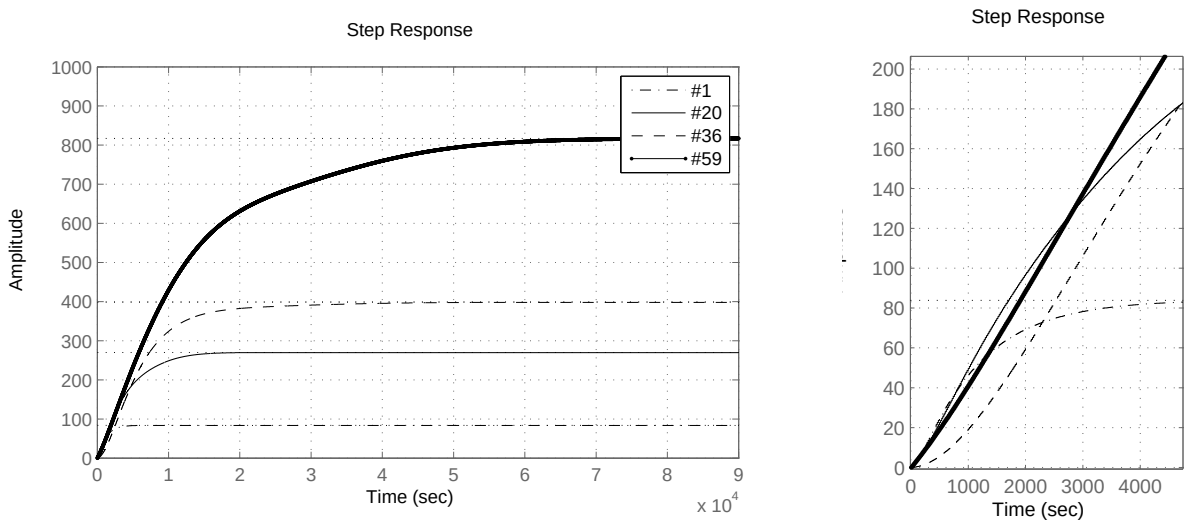

Fig. 2.2: Left: Step response for different models, Right: Zoomed view of step responses showing the start up time T_u

Open loop step responses of some representative identified models are shown in Fig. 2.2. The models represent a slow system with a large gain K. Differences between the models can be seen in

their slope S, rise time T_R, static gain K_∞ and start up time T_U (time needed to reach 0.5% of the final gain).

By comparing the static gain K_∞ ($K_\infty \approx 80 \ldots 800$) to the setpoint $y_r = 0.9$ one can see that only small amounts of drug will be needed to relax the muscles. The rise time T_R of the system indicates the slow dynamics of the identified models. This implies that the controller has to make the closed loop much faster than the open loop in order to achieve the control objective.

3 Hybrid Observer

The hybrid observer is the part of the closed-loop system that identifies the model which matches the patient best in each time step. It is assumed that the dynamics of the real system are represented by a true model $\mu(n)$ which produces the measured output at time step n (see Equ. (2.2)). Therefore the model identification is equivalent to the calculation of a discrete state estimate $\hat{\mu}(n)$. The hybrid observer must additionally be able to calculate a continuous state estimate $\hat{x}_k(n)$ for every model k. Both information will be used by the switched controller to calculate the optimal input $u(n)$ to the patient.

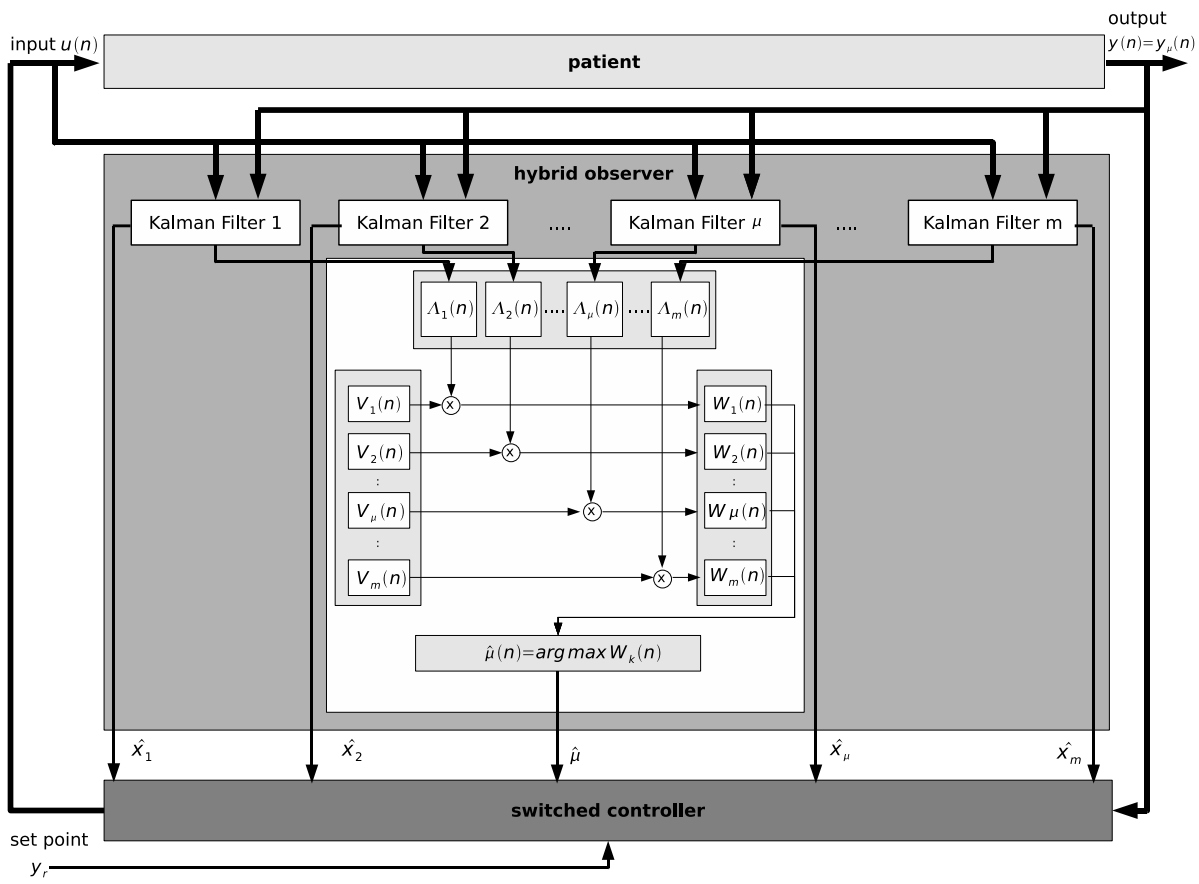

Fig. 3.1: Hybrid observer (schematic)

Since little research has been done in the field of hybrid observation, different techniques will be compared in this section to find the observer most suitable for the proposed problem. All algorithms compared in this section are based on the “Multiple Model Adaptive Estimation” (MMAE) algorithm and will be introduced in section 3.1 - 3.2. The MMAE algorithm uses m different Kalman filters to calculate a likelihood Λ_k for

each model to match the patient based on its system dynamics. Additional information about the switching process is used to calculate a likelihood V_k of the model to match the patient based on the overall discrete dynamics. Both values are combined in the likelihood W_k to calculate how well the different models match the controlled patient. Based on that information the best matching model $\hat{\mu}(n)$ is selected (see Fig. 3.1).

3.1 Kalman Filter

The hybrid observer uses m Kalman filter to observe the system dynamics of the different modes. The following section gives a brief review of Kalman filters. To simplify the equations, only one patient model is used in this section ($m = 1$, $k = 1$).

As shown in chapter 2, the patient model can be understood as a discrete-time, time-invariant linear plant that is driven by white process noise. Furthermore, it is assumed that the measurement is corrupted by white noise and the input is known and deterministic.

$$\begin{aligned} x(n+1) &= Ax(n) + Bu(n) + \Gamma w(n) & E\{w(n)w^T(n)\} &= Q(n) \\ y(n) &= Cx(n) + v(n) & E\{v(n)v^T(n)\} &= R(n) \end{aligned} \tag{3.1}$$

In (3.1) w and v are the process and measurement noises respectively and Q and R are the corresponding covariances. Both disturbances are zero-mean white Gaussian sequences which are mutually independent. The process noise Q is assumed to be zero because all calculations are based on simulations. For the measurement noise R the standard deviation $\sigma = 0.3$ of the output y is used (see section 2.2).

In every time step the Kalman filter calculates a first estimate $\hat{\tilde{x}}(n)$ of the current state and the covariance $\tilde{P}(n)$ of the prediction error $\tilde{e}(n)$ based on the patient model shown in (3.1). This is done with the recursive formulas (3.2) which use the updated state estimate $\hat{x}(n-1)$ and the updated covariance $P(n-1)$ of the previous time step.

$$\begin{aligned} \hat{\tilde{x}}(n) &= A\hat{x}(n-1) + Bu(n-1) \\ \tilde{P}(n) &= E\{\tilde{e}(n)\tilde{e}^T(n)\} = AP(n-1)A^T + \Gamma Q(n-1)\Gamma^T \qquad \tilde{e}(n) = x(n) - \hat{\tilde{x}}(n) \end{aligned} \tag{3.2}$$

Based on the first state estimate $\hat{\tilde{x}}(n)$ the output residual $r(n)$ and its associated covariance $S(n)$ are calculated.

$$\begin{aligned} r(n) &= y(n) - \hat{y}(n) = y(n) - C\hat{\tilde{x}}(n) \\ S(n) &= E\{r(n)r^T(n)\} = C\tilde{P}(n)C^T + R(n) \end{aligned} \tag{3.3}$$

Using the measurement $y(n)$, the state estimate and the covariance are updated.

$$\hat{x}(n) = \hat{\tilde{x}}(n) + K(n)r(n)$$
$$P(n) = (I - K(n)C)\tilde{P}(n) \tag{3.4}$$

The Kalman Gain $K(n)$ involved in this process is designed to make the state estimate optimal.

$$K(n) = \tilde{P}(n)C^T \left(C\tilde{P}(n)C^T + R(n)\right)^{-1} \tag{3.5}$$

The MMAE algorithm introduced in the following section will use the residual $r(n)$ and its associated covariance $S(n)$ of every model.

3.2 Multiple Model Adaptive Estimation (MMAE)

The basic idea of the MMAE algorithm is to combine information about the m different linear systems and the discrete process to calculate the best matching model $\hat{\mu}(n)$ in every time step. As mentioned in the previous section, the residuals $r_k(n)$ and their associated covariances $S_k(n)$ are provided by the Kalman filters. Those values are used to calculate the probability $\Lambda_k(n)$ of model k to match the patient based on its system dynamics. The basic idea behind this is the inverse relation of the residual $r_k(n)$ to the best matching model. If the measurement noise would only be white (what is assumed but not possible in reality) the true model would generate a residual $r_k(n) = 0$. Therefore the probability $\Lambda_k(n)$ is calculated as a normal Gaussian distribution with mean value $\xi = 0$ and covariance $\sigma^2 = S_k(n)$.

$$\Lambda_k(n) = \mathcal{N}(r_k(n); 0; S_k(n)) = \frac{1}{\sqrt{2\pi S_k(n)}} e^{-\frac{r_k^2(n)}{2S_k(n)}}$$

This probability $\Lambda_k(n)$ is maximal if $r_k(n) = 0$ indicating a perfect match between the patient and the model. Additionally the MMAE calculates the probability $V_k(n)$ for every model to match the patient based on information about the switching scheme. Since all observation algorithms compared in this section vary in the calculation of this second probability, the different methods to estimate V_k are presented in section 3.2.1. The final probability $W_k(n)$ of model k to match the patient is calculated by multiplying both probabilities and normalizing them.

$$W_k(n) = \frac{\Lambda_k(n)V_k(n)}{\sum_{j=1}^{m} \Lambda_j(n)V_j(n)} \tag{3.6}$$

After the likelihood $W_k(n)$ is calculated for every model, the discrete state estimate $\hat{\mu}(n)$ is calculated by identifying the model with the highest probability.

$$\hat{\mu}(n) = arg \max_k W_k(n)$$

The schematic representation of this algorithm can be seen in Fig. 3.1.

3.2.1 Calculation of $V_k(n)$

The hybrid observers compared in this work use two different approaches to calculate the likelihood $V_k(n)$. The first was published by S. Fekri, M. Athans and A. Pascoal in [1] and uses no assumptions about the switching scheme. This algorithm was explicitly invented for model switched adaptive control. The second method uses a Markov process to model switches between the different models. This strategy was introduced by I. Hwang, H. Balakrishnan and C. Tomlin in [2] to track the flight mode of aircrafts.

No Assumptions about Switching Scheme

This method to calculate $V_k(n)$ was presented in [1] and uses no assumptions about the switching scheme. It is assumed that the essential information about the discrete dynamics is the evolution of the probabilities W_k for every model over time. Considering that abrupt changes in the true model μ are unlikely to occur, a change in μ would be indicated by changing probabilities W_μ. That means that an unlikely model of time step $n-1$ will be unlikely to match perfectly in time step n and vice versa. Therefore, the probability

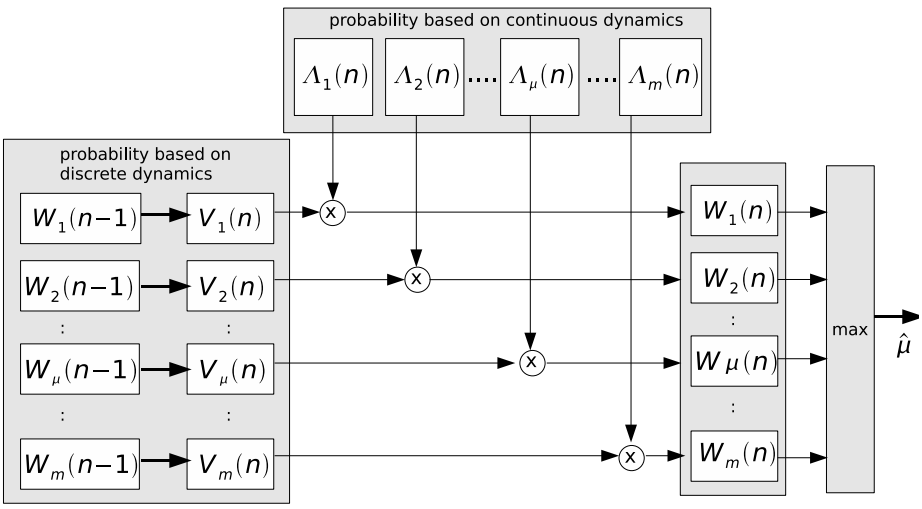

Fig. 3.2: Calculation of Probability $W_k(n)$

$W_k(n-1)$ of model k to match the patient in the previous time step is used as $V_k(n)$ (see Fig. 3.2).

$$V_k(n) = W_k(n-1)$$

This changes the final probability $W_k(n)$ in (3.6) to

$$W_k(n) = \frac{\Lambda_k(n) W_k(n-1)}{\sum_{j=1}^{m} \Lambda_j(n) W_j(n-1)} \tag{3.7}$$

The algorithm is initialized with $W_k(0) = \frac{1}{m}$.

Stochastic Switching Scheme

This method to calculate $V_k(n)$ was presented in [2] and models the mode transitions as a Markov process, shown in (3.8). The transition matrix Π contains the probabilities π_{ij} to switch from mode j to mode i and its column sum is one ($\sum_{i=1}^{m} \pi_{ij} = 1, \forall j$).

$$\mu(n+1) = \Pi\mu(n) \tag{3.8}$$

The calculation of the probabilities π_{ij} is based on additional patient information (e.g. gender, race or weight) leading to a formation of model subgroups in the data set. If the controlled patient belongs to one subgroup, models in this subgroup are more likely to be the best match. This leads to a Markov matrix Π with nonzero entries (i.e. theoretically every model can match the patient) and higher values for transitions into the selected subgroup.

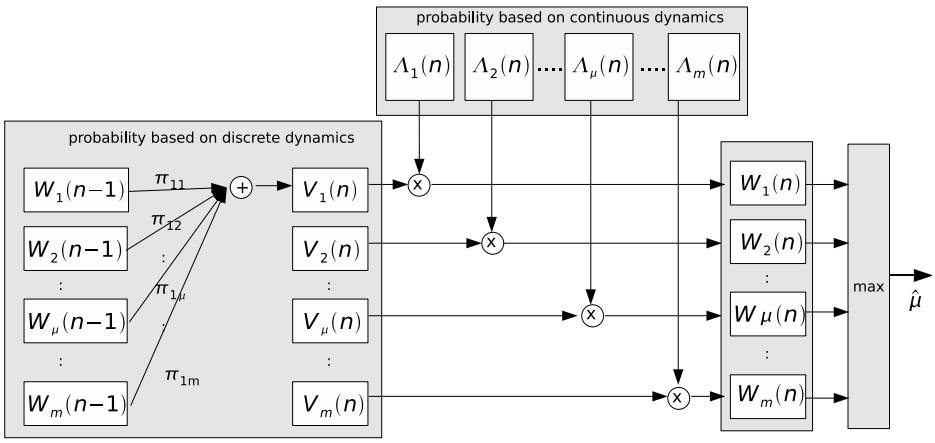

Fig. 3.3: Calculation of Probability $W_k(n)$

To use this additional information, $V_k(n)$ is calculated as the sum over joint probabilities $v_j(n)$ (see Fig. 3.3). These probabilities $v_j(n)$ combine the probability $W_j(n-1)$ of model j to match the patient in the previous time step and the transition probability π_{kj} to switch from mode j to mode k. The use of the probabilities $W_j(n-1)$ in $V_k(n)$ can be argued in the same way as presented in section 3.2.1.

$$\begin{aligned} v_j(n) &= \pi_{kj} W_j(n-1) \\ V_k(n) &= \sum_{j=1}^{m} v_j(n) = \sum_{j=1}^{m} \pi_{kj} W_j(n-1) \end{aligned} \tag{3.9}$$

This changes the final probability $W_k(n)$ in (3.6) to

$$W_k(n) = \frac{\Lambda_k(n) \sum_{i=1}^{m} \pi_{ki} W_i(n-1)}{\sum_{j=1}^{m} \left[\Lambda_j(n) \sum_{i=1}^{m} \pi_{ji} W_i(n-1)\right]} \tag{3.10}$$

The algorithm is initialized with $W_k(0) = 1/m$.

3.2.2 Improvements

Some improvements to the discrete state estimation were suggested in [2]. Those improvements will be introduced and their influences on the hybrid observers in muscle relaxation will be evaluated.

Coupling Kalman Filters

The idea behind the coupling of Kalman filters is to use the likelihoods W_k to improve the observation of the linear process. Therefore the state estimates $\hat{x}_k(n)$, $\forall k$ are updated in every time step by using a mixing probability $\nu_{ki}(n)$.

$$\nu_{ki}(n) = \frac{\pi_{ki} W_i(n)}{\sum_{k=1}^{m} \pi_{ki} W_i(n)}$$

The updated state estimate $\hat{x}_{0,k}(n)$ will then be used by the Kalman filter in (3.2) to calculate the first state estimate of time step $k+1$. It is assumed that the Kalman filter converges better if more information about the process are provided. Since the covariance $P_k(n)$ is connected to the state estimate, it has to be updated as well to ensure the Kalman filter is working optimally. (note: $*$ stands for element-wise multiplication)

$$\begin{aligned} \hat{x}_{0,k}(n) &= \sum_{i=1}^{m} \hat{x}_i(n) * \nu_{ki} \\ P_{0k}(n) &= \sum_{i=1}^{m} \left[P_i(n) + \left(\hat{x}_i(n) - \hat{x}_{0,k}(n)\right) \left(\hat{x}_i(n) - \hat{x}_{0,k}(n)\right)^T * \nu_{ki} \right] \end{aligned} \tag{3.11}$$

Residual Mean

The second improvement considered in this work is to use the mean value of all residuals from the start to the current time step of the current model $\bar{r}_k(n)$

$$\bar{r}_k(n) = \sqrt{\frac{\sum_{j=1}^{n} r_k^2(j)}{n}}$$

It was shown in [2] that this leads to an improvement of the performance of the MMAE.

3.2.3 Algorithms

Based on the algorithms discussed in section 3.2.1 and the improvements shown in section 3.2.2, a variety of different hybrid observation algorithms can be formulated

B-MMAE **B**asic - **M**ultiple **M**odel **A**daptive **E**stimation
- without any knowledge about the switching

$$W_k(n) = \frac{\Lambda_k(n) W_k(n-1)}{\sum_{j=1}^{m} \Lambda_j(n) W_j(n-1)}$$

B-RM-MMAE **B**asic - **R**esidual **M**ean - **M**ultiple **M**odel **A**daptive **E**stimation
- B-MMAE with mean values of residuals $\bar{r}_k$

S-MMAE **S**tochastic - **M**ultiple **M**odel **A**daptive **E**stimation
- stochastic switching scheme

$$W_k(n) = \frac{\Lambda_k(n) \sum_{i=1}^{m} \Pi_{ki} W_i(n-1)}{\sum_{j=1}^{m} [\Lambda_j(n) \sum_{i=1}^{m} \Pi_{ji} W_i(n-1)]}$$

S-RM-MMAE **S**tochastic - **R**esidual **M**ean - **M**ultiple **M**odel **A**daptive **E**stimation
- S-MMAE with mean values of residuals $\bar{r}_k$

S-I-MMAE **S**tochastic - **I**nteractive - **M**ultiple **M**odel **A**daptive **E**stimation
- S-MMAE with coupling of Kalman Filters

S-RM-I-MMAE **S**tochastic - **R**esidual **M**ean - **I**nteractive - **M**ultiple **M**odel **A**daptive **E**stimation
- S-MMAE with:
  - coupling of Kalman Filters
  - mean values of residuals $\bar{r}_k$

3.3 Simulations

In order to compare the different algorithms outlined above simulations needed to be done. For this purpose one model k was randomly selected to be the true model μ. It was used to generate the output y as its open loop step response. Based on this data the hybrid estimation algorithms were run. It was observed how fast and how well they were able to track the true model μ. As a result of the open loop analysis discussed in section 4.2.1, all models were reordered and numbered according to their open loop start up time T_u (time needed to reach 0.5% of the final gain).

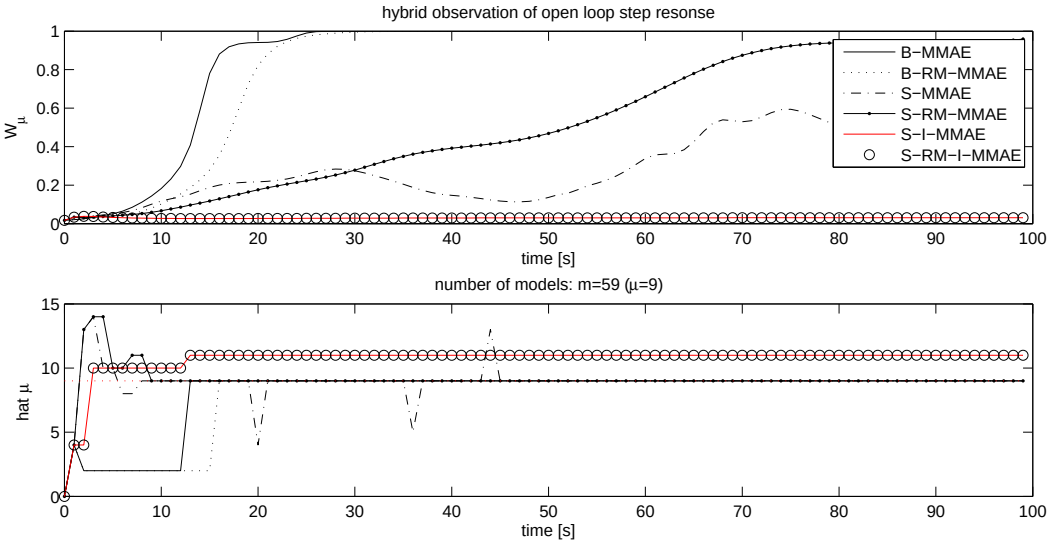

Fig. 3.4: Number of models: $m = 59$, true model: $\mu = 9$

All plots of those simulations show the evaluation of the true model likelihood W_μ in the upper graph and the discrete state estimate $\hat{\mu}$ in the lower graph. To stress the differences between the models, different studies have been performed. Because the simulation with all $m = 59$ models (see Fig. 3.4) showed some performance weaknesses, it was tested if those algorithms will improve by using subgroups of the models. Therefore Fig. 3.5 and Fig. 3.6 show the simulation of patient subgroups selected by picking every k_s model. As a result those subgroups contain models which are well differentiable in terms of their open loop behavior. In contrary Fig. 3.7 and Fig. 3.8 show simulations with a subgroup of models with similar step responses. These were constructed by selecting a certain number of models around the randomly picked true model μ for simulation.

By looking at Fig. 3.4 - 3.8 it is obvious that both interactive strategies (S-I-MMAE and S-I-RM-MMAE) give the same estimates and are not able to track the true discrete state in any scenario. It was seen during implementation that the number of numerical problems, arising by coupling the Kalman filters, increases with an increasing number of models. Since the algorithm was introduced and tested in [2] by using only two different modes, it is assumed that the number of modes used in the muscle relaxation problem is not suitable for this particular estimation technique.

The other four tested algorithms show sufficiently fast convergence to the true mode and are able to track the true mode μ in all scenarios. It can be seen, that the B-MMAE and the B-RM-MMAE as well as the S-MMAE and the S-RM-MMAE are behaving similar respectively. The use of the residual mean improves

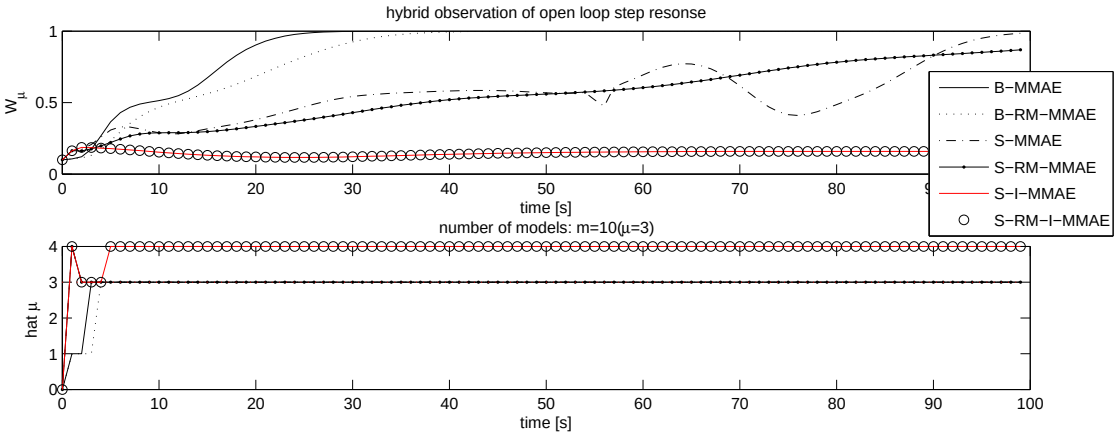

Fig. 3.5: Number of models: $m = 10$, true model: $\mu = 3$

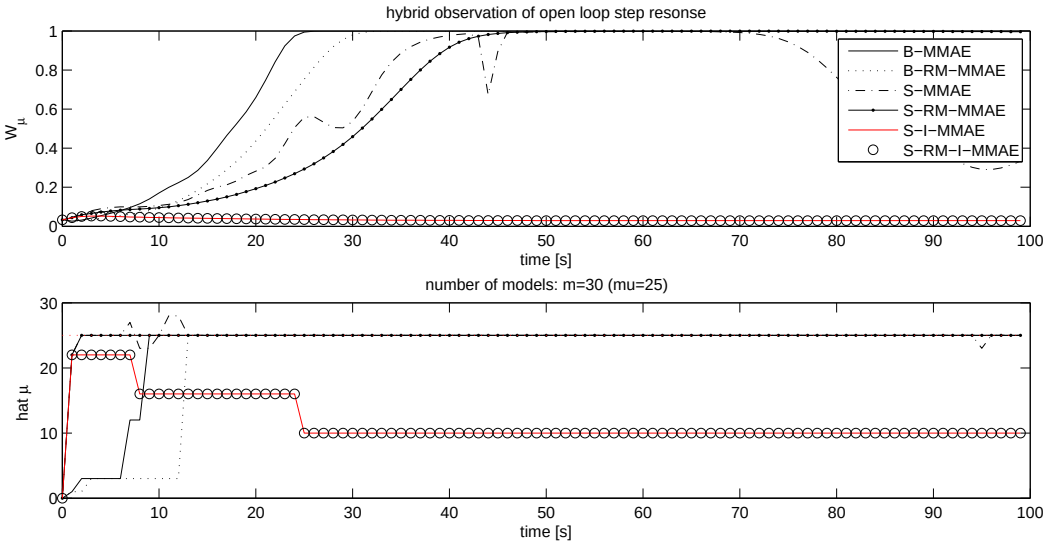

Fig. 3.6: Number of models: $m = 30$, true model: $\mu = 25$

the performance of the S-MMAE but does not improve the performance of the B-MMAE. One main difference between the basic and the stochastic algorithms can be seen by looking at the convergence of the likelihood W_μ to one. The graphics show that the basic algorithm will converge after about 20 samples in each scenario. The convergence of the stochastic algorithm is dependent on the size of the subgroup and on the true model μ. The second difference between these algorithms is the settling time of their discrete state estimate. Even though only the stochastic algorithm shows an effect of different subgroup sizes on the likelihood W_μ, it can be seen that both algorithms converge faster if the subgroup is smaller. Usually the stochastic algorithm picks a model with a larger start up time T_u first and converges fast to the true model μ. The basic algorithm converges slower, and usually picks a fast model in the beginning. These two properties will have an influence on the closed-loop behavior as will be shown in chapter 5.

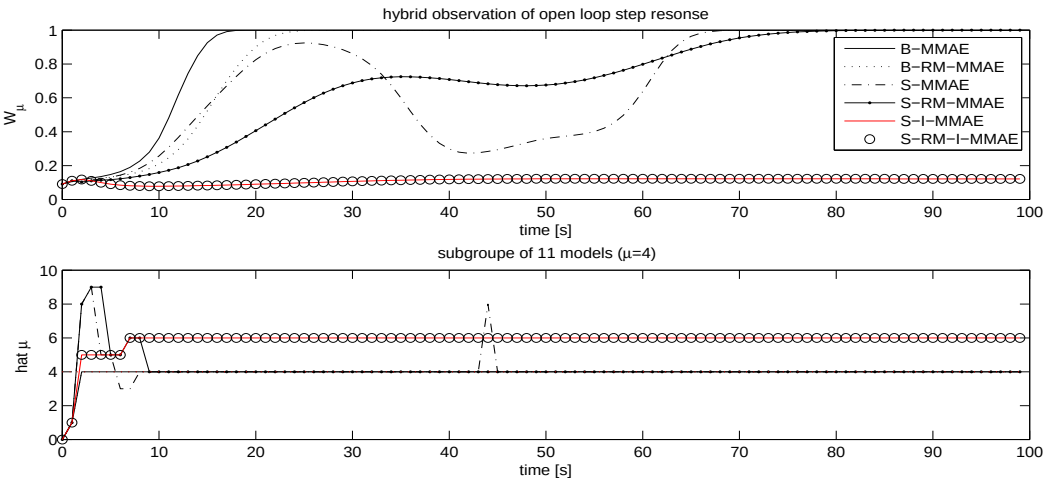

Fig. 3.7: Number of models: $m = 11$, true model: $\mu = 4$

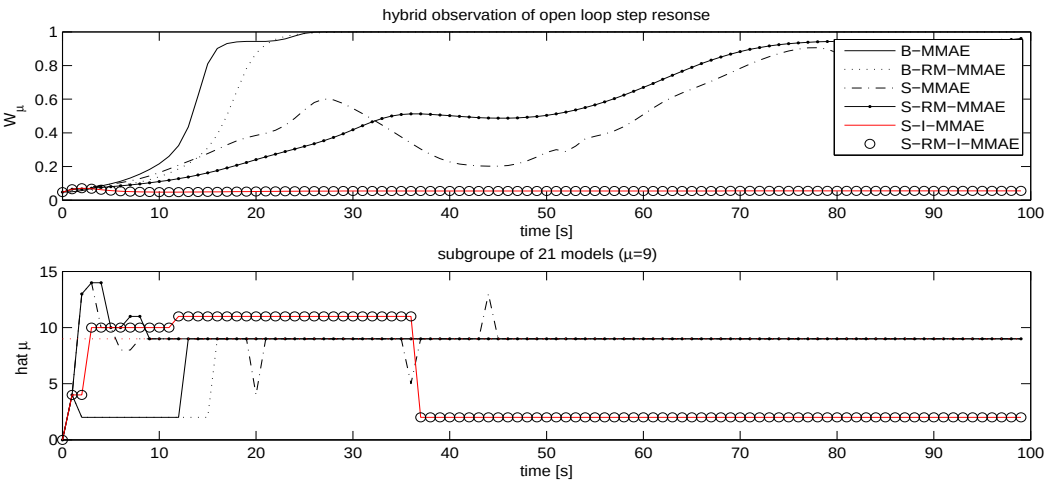

Fig. 3.8: Number of models: $m = 21$, true model: $\mu = 9$

proportional

As a conclusion to this observation, the interacting algorithms will not be used for simulations of the final closed-loop system. Additionally the use of the residual mean does not improve the performance of the B-MMAE algorithm but does make the stochastic algorithm significantly more stable. Therefore, the B-MMAE and the S-RM-MMAE have proved to be the most reliable hybrid observation techniques tested in this work. The influence of their differences on the closed-loop performance will be investigated with closed-loop simulations in chapter 5.

4 Switched Controller

This part of the closed-loop system consists of m different controllers, each one tuned for one specific model. Based on the discrete state estimate $\hat{\mu}(n)$ the controller associated with model μ is selected and its output $u(n)$ is used as the input to the system (see Fig. 4.1).

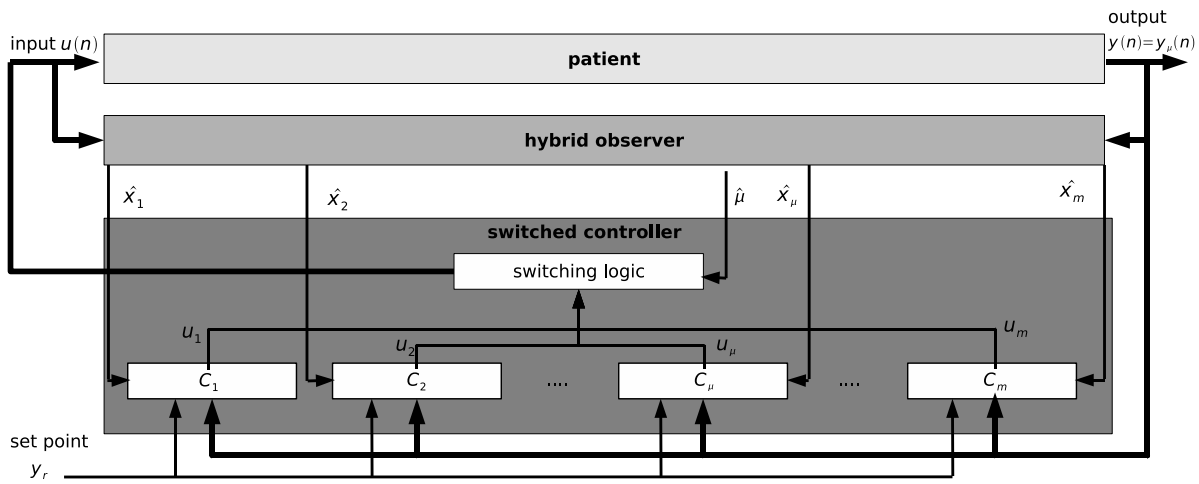

Fig. 4.1: Switched controller (schematic)

It was shown in section 2.3 that these controllers have to make the closed loop much faster than the open loop in order to achieve the control objective. It would be also advantageous if the controller could handle input constraints, since the infusion rate of Rocuronium has a lower and upper bound. Furthermore, the measurement is discretized with a fairly large time step of 20 seconds as shown in section 2.2. These facts indicate that the use of a Model Predictive Controller can be useful to handle the given problems. It was also shown in [3] that a Generalized Predictive Controller (GPC) is suited well for controlling the proposed system. Therefore this control technique was chosen.

To introduce this control algorithm, the concept of Generalized Predictive Control will be explained first. To simplify the equations this will only be done for a single controller ($m = 1$). Afterwards the tuning of the m different GPC will be explained and their performance will be shown with the help of simulations.

4.1 Generalized Predictive Control (GPC)

As a special Model Predictive Controller, the GPC uses a model of the plant to predict responses of the system to a given control sequence. This prediction is used to search for an optimal future input sequence to archive the given control objective. In the proposed problem, the goal is to minimize the sum $\sum_j e(n+j)$ of the error $e(n+j)$ between the setpoint y_r and the output y without using to much control action. This goal can be formulated in a performance index J shown in (4.1), suggested for example in [7]

$$\begin{aligned} J &= E\left[\sum_{j=1}^{h_u}\left[y(n+j)-y_r(n+j)\right]^2+\sum_{j=1}^{h_c}\rho\left[\Delta u(n+j-1)\right]^2\right] \\ &= \left[\sum_{j=1}^{h_u}\left[\hat{y}(n+j)-y_r(n+j)\right]^2+\sum_{j=1}^{h_c}\rho\left[\Delta u(n+j-1)\right]^2\right] \\ &= \left[(\hat{Y}-Y_r)^T(\hat{Y}-Y_r)+(\Delta U)^T\rho(\Delta U)\right] \end{aligned} \tag{4.1}$$

The GPC minimizes this index J with respect to an input sequence $u(n+j)$, $j=1\ldots h_c$. The length of the input sequence is given by the control horizon h_c. For every timestep $j>h_c$ the control input $u(n+j)$ is zero. The number of predicted outputs $\hat{y}(n+j)$, $j=1\ldots h_u$ used in (4.1) is restricted by the maximal prediction horizon $h_u>h_c$. The factor ρ is a control weighting factor which ensures physical realizability of the input sequence. A minimal prediction horizon h_l can be introduced to handle time delays. Then the sequence of outputs considered in (4.1) is forced to start at time h_l. This accounts for a certain time period $h_l\cdot T_s$ needed by the system to respond to the input sequence. Using h_l in (4.1) gives

$$J = E\left[\sum_{j=h_l}^{h_u}(y(n+j)-y_r(n+j))^2+\sum_{j=1}^{h_c}\rho(\Delta u(n+j-1))^2\right] \tag{4.2}$$

Laguerre models usually do not have explicit time delays as discussed in section 2.1. Therefore, a minimum prediction horizon is usually not introduced when using this kind of models. Nevertheless it will be shown in section 4.2.2 that this forth parameter is very useful for tuning in muscle relaxation control. But to simplify the equations, we assume $h_l=1$ in this section (the equations can similarly be used for $h_l>1$). The vectors $\hat{Y}$, Y_r and ΔU in (4.1) are then defined as

$$\hat{Y}=\begin{pmatrix}\hat{y}(n+1)\\ \hat{y}(n+2)\\ \vdots\\ \hat{y}(n+h_u)\end{pmatrix} \qquad Y_r=\begin{pmatrix}y_r(n+1)\\ y_r(n+2)\\ \vdots\\ y_r(n+h_u)\end{pmatrix} \qquad \Delta U=\begin{pmatrix}u(n)-u(n-1)\\ u(n+1)-u(n)\\ \vdots\\ u(n+h_c)-u(n+h_c-1)\end{pmatrix}$$

The components of the output prediction $\hat{Y}$ are calculated as shown in (4.3).

$$\begin{aligned}\hat{y}(n+j) &= C\hat{x}(n+j) + \epsilon(n+j) \qquad\qquad \epsilon(n+j) \approx \epsilon(n) = y(n) - \hat{y}(n)\\ &= C\hat{x}(n+j) + (y(n) - \hat{y}(n))\\ &= y(n) + C(\hat{x}(n+j) - \hat{x}(n)) \end{aligned} \tag{4.3}$$

The output prediction error $\epsilon(n+j)$ is assumed to be approximately equal to the current prediction error $\epsilon(n+j)$. This leads to " [...] a closed-loop prediction which guarantees unbiased prediction in the presence of unstructured uncertainties" (Dumont [7], p.1914). The predicted future states $\hat{x}(n+j)$ in (4.3) are calculated based on the current state estimate $\hat{x}(n)$ of the Kalman Filter and the future input sequence $u(n+j)$ $j = 0...h_c$.

$$\begin{aligned}\hat{x}(n+1) &= A\hat{x}(n) + Bu(n)\\ \hat{x}(n+2) &= A\hat{x}(n+1) + Bu(n+1) \quad = \quad A^2\hat{x}(n) + ABu(n) + Bu(n+1)\\ \vdots \quad &= \qquad\qquad \vdots\\ \hat{x}(n+j) &= A^j\hat{x}(n) + A^{j-1}Bu(n) + A^{j-2}Bu(n+1) + \cdots + Bu(n+j-1) \end{aligned} \tag{4.4}$$

It follows from substituting (4.4) in (4.3) that

$$\begin{aligned}\hat{y}(n+j) =& y(n) + C\left[A^j\hat{x}(n) + \left[A^{j-1}u(n) + A^{j-2}u(n+1) + \ldots + u(n+j-1)\right]B - \hat{x}(n)\right]\\ =& \underbrace{y(n) + C(A^j - I)\hat{x}(n)}_{l_j} + C\left[A^{j-1}u(n) + A^{j-2}u(n+1) + \ldots + u(n+j-1)\right]B \end{aligned} \tag{4.5}$$

In matrix form (4.5) becomes

$$\begin{pmatrix}\hat{y}_{(n+1)}\\ \hat{y}_{(n+2)}\\ \vdots\\ \hat{y}_{(n+h_u)}\end{pmatrix} = \begin{pmatrix}l_1\\ l_2\\ \vdots\\ l_{h_u}\end{pmatrix} + \begin{pmatrix}CB & 0 & \ldots & 0\\ CAB & CB & \ldots & 0\\ & \ddots & \ddots & \\ CA^{h_u-1}B & CA^{h_u-2}B & \ldots & CA^{h_u-h_c-1}B\end{pmatrix}\begin{pmatrix}u_{(n)}\\ u_{(n+1)}\\ \vdots\\ u_{(n+h_c)}\end{pmatrix} \tag{4.6}$$

Equation (4.6) can now be regrouped into know and unknown terms at time $t(n)$ by separating the free and the forced response of the system. The free response f is the output signal $y(n+j)$ if the present and future control signals are zero ($u(n+j) = 0\ \forall j \geq 0$) whereas the forced response is the response of the system due to future control actions.

$$\begin{pmatrix} \hat{y}_{(n+1)} \\ \hat{y}_{(n+2)} \\ \vdots \\ \hat{y}_{(n+h_u)} \end{pmatrix} = \underbrace{\underbrace{\begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_{h_u} \end{pmatrix}}_{L} + \underbrace{\begin{pmatrix} CB \\ CB + CAB \\ \vdots \\ \sum_{i=0}^{h_u-1} CA^iB \end{pmatrix}}_{S_1} u(n-1)}_{\text{free response } (F)}$$

$$+ \underbrace{\underbrace{\begin{pmatrix} CB & 0 & \dots & 0 \\ CB + CAB & CB & \dots & 0 \\ & \ddots & \ddots & \\ \sum_{i=0}^{h_u-1} CA^iB & \sum_{i=0}^{h_u-2} CA^iB & \dots & \sum_{i=0}^{h_u-h_c} CA^iB \end{pmatrix}}_{S} \underbrace{\begin{pmatrix} u(n) - u(n-1) \\ u(n+1) - u(n) \\ \vdots \\ u(n+h_c) \\ -u(n+h_c-1) \end{pmatrix}}_{\Delta U}}_{\text{forced response}} \tag{4.7}$$

In (4.7) $F = [f_1 f_2 \dots f_{h_u}]^T$ is the vector of the free responses and $S = [S_1 S_2 \dots S_{h_u}]$ is a lower triangular matrix of the step response coefficients where S_i is the ith column of S. This leads to the final formula for the output prediction vector $\hat{Y}$:

$$\hat{Y} = L + S_1 u(n-1) + S\Delta U = F + S\Delta U \tag{4.8}$$

This expression can now be used to calculate the performance index for the given control problem. As one can see by substituting (4.8) into (4.1) the cost function can be rewritten as

$$\begin{aligned} J &= \left[(F + S\Delta U - Y_r)^T (F + S\Delta U - Y_r) + \Delta U^T \rho \Delta U\right] \\ &= \underbrace{\Delta U^T (S^T S + \rho I)\Delta U + 2\Delta U^T S^T (F - Y_r)}_{J_1} + (F - Y_r)^T (F - Y_r) \end{aligned} \tag{4.9}$$

and will be minimal if J_1 is minimal with respect to ΔU

$$\min_{\Delta U} J_1 = \min_{\Delta U} \left[\Delta U^T (S^T S + \rho I)\Delta U + 2\Delta U^T S^T (F - Y_r)\right] \tag{4.10}$$

were (4.10) shows a Quadratic Programming (QP) optimization problem

$$\min_x \left[\frac{1}{2} x^T H x + g^T x\right] \tag{4.11}$$

with $x = \Delta U$, $H = 2(S^T S + \rho I)$ and $g = 2S^T (F - Y_r)$. After the future optimal input sequence is obtained by solving (4.10), its first component is used to calculate the current control input $u(n)$.

$$u(n) = \Delta u + u(n-1)$$

In the next time step the whole procedure is redone and the next optimal input is calculated.

The major advantage of this method is its ability to actively handle constraints. This leads to a reformulation of (4.10) into a constrained QP problem were the optimal input is calculated with respect to those constraints. A drawback of this method is that the QP problem can only be solved by using numerical optimization methods which introduces several problems. First, no analytical expression for the controller can be given if $u(n)$ is calculated numerically. This makes the analysis of the controller, for example regarding stability or convergence, very difficult. Another problem is that solving a constrained QP problem in every time step can be very computationally expensive.

Therefore, the introduced GPC was changed to a more practical version that uses a control horizon of one ($h_c = 1$). This means that the optimization algorithm assumes that the future input changes only at time $t(n)$ and will be constant for $t(n+j)$. This leads to

$$\begin{aligned} u(n+j) &= u(n) \qquad \forall j \geq 1 \\ \Delta u(n+j) &= 0 \qquad \forall j \geq 1 \end{aligned}$$

Therefore (4.7) simplifies to

$$\hat{Y} = F + S_1 \Delta u(n) \tag{4.12}$$

and (4.10) becomes

$$\min_{\Delta u(n)} J_1 = \min_{\Delta u(n)} \left[(S_1^T S_1 + \rho) \Delta u^2(n) + 2\Delta u(n) S_1^T (F - Y_r) \right] \tag{4.13}$$

Since a control horizon of $h_c = 1$ leads to a one dimensional performance index, equation (4.13) can be solved by setting its derivative to zero

$$\frac{\partial J_1}{\partial \Delta u(n)} = 2(S_1^T S_1 + \rho)\Delta u(n) + 2S_1^T (F - Y_r) \stackrel{!}{=} 0$$

leading to

$$\Delta u(n) = (S_1^T S_1 + \rho)^{-1} S_1^T (Y_r - F) = u(n) - u(n-1)$$

$$\begin{aligned} u(n) &= (S_1^T S_1 + \rho)^{-1} S_1^T (Y_r - F) + u(n-1) \\ &= (S_1^T S_1 + \rho)^{-1} \left[S_1^T Y_r - S_1^T L + \rho u(n-1) \right] \end{aligned} \tag{4.14}$$

By simplifying the GPC and setting h_c to one, the controller looses the ability to handle input constraints. Therefore the input is set to zero if $u(n) < 0$.

4.2 Controller Tuning

The concept of the switched controller used in this work, is to have an optimally tuned controller for each patient model. By using the database of models collected in [3], this involves the tuning of 59 controllers. Furthermore the future goal of this work is to collect more data leading to new models to be included in the database. Therefore it is reasonable to develop a tuning routine for the introduced GPC based on the open loop performance of a model and the closed-loop requirements for the system.

As mentioned in chapter 1 the control objective for this project is to make the closed loop fast while keeping the time periods of saturation as short as possible. This leads to the following closed-loop constraints:

overshoot $\leq 10\%$

rise time ≤ 10 minutes

It will be investigated which parameters of the GPC should be used to tune the controllers for those constraints. Afterwards two different tuning algorithms are introduced and their performance is compared. Finally closed-loop simulations of the single models with their optimal tuned controller are shown.

4.2.1 Open Loop Analysis

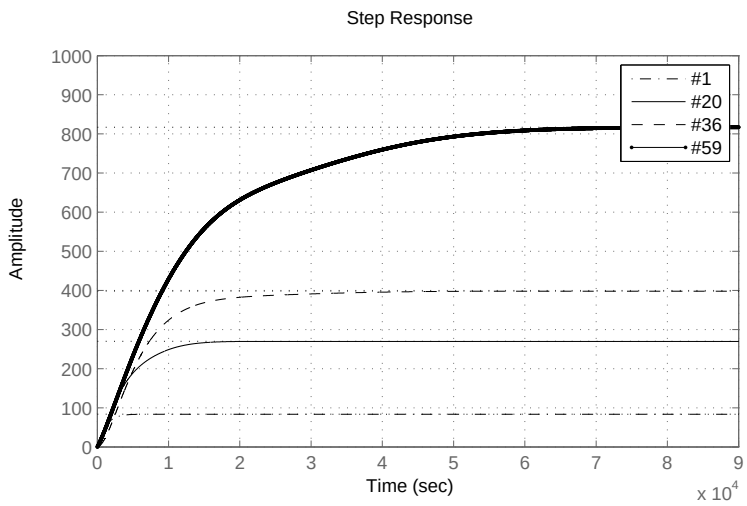

Fig. 4.2: Step response for different models

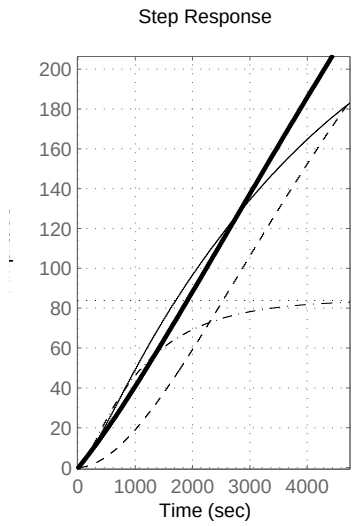

Fig. 4.3: Detail

The step responses in Fig. 4.2 and the bode plot in Fig. 4.4 show, that the patient models vary in their

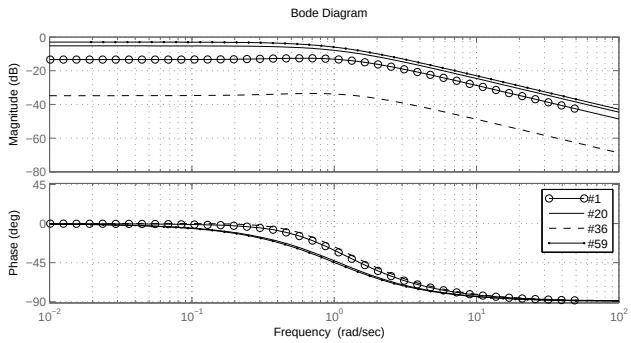

Fig. 4.4: Bode plot of different models

S slope

T_R rise time

K_∞ static gain

T_U start up time (time needed to reach 0.5% of the final gain)

K gain

In order to develop a tuning routine these open loop parameters are tested for correlations with the tuning parameters of the GPC. It will be investigated, if the optimal tuning parameters are correlated with the open loop behavior of each model.

4.2.2 Closed Loop Analysis

The GPC for Laguerre models has three different tuning parameters mentioned in section 4.1.

h_u maximum prediction horizon

ρ control weighting factor

h_c control horizon

Additionally one can see in Fig. 4.2 that all models are very slow. Between 1×10^4 to 6×10^4 seconds are needed to reach the steady state compared to the desired rise time in the closed-loop system of 900 seconds ($= 15$ minutes). This behavior can be interpreted as a dead time equivalent. With a GPC, dead times are handled by using a minimum prediction horizon h_l as shown in (4.2). Therefore a forth tuning parameter will be used:

h_l minimum prediction horizon

Therefore four tuning parameters can be used to optimize the control performance for every model. As discussed in section 4.1, the control horizon h_c will be set to one. If the minimum prediction horizon h_l is used for tuning it is furthermore convenient to introduce a new parameter $d = h_u - h_l$, where d is called distance. This parameter replaces h_u in the further description. Since the GPC is based on an optimization over a certain amount of time steps in the future, the distance d is the most important parameter for the controller and needs to be tuned.

To verify the influence of ρ and h_l on the closed-loop behavior of the system, a slow model ($\#1$) and a fast model ($\#59$) were tuned with different ρ and h_c. This was done by holding one tuning parameter constant while the other one was varied over time. Tab. 4.1 shows that ρ has no significant influence on the rise time nor the overshoot. On the contrary, Tab. 4.2 shows the strong influence of h_l on the closed-loop performance of the two chosen models. Therefore the latter was used in combination with the distance d to tune the 59 patient models. Varying d will change the number of predicted outputs used in the performance index J and varying h_l shifts this window of predicted outputs back and forth in time. The control weighting factor ρ will be set to zero for all further simulations.

			model $\#1$		model $\#59$	
d	h_l	ρ	rise time (T_R)	overshoot	rise time (T_R)	overshoot
10	1	0.0	15.509209	1.050939	19.626359	19.137556
10	1	0.8	15.503514	1.053000	19.617628	19.145494
10	1	1.6	15.497807	1.055060	19.608878	19.153438
10	1	2.4	15.492090	1.057121	19.600107	19.161390
10	1	3.2	15.486363	1.059182	19.591317	19.169348
10	1	4.0	15.480624	1.061242	19.582506	19.177314
10	1	4.4	15.477750	1.062272	19.578094	19.181300
10	1	5.2	15.471996	1.064333	19.569253	19.189276
10	1	6.0	15.466230	1.066393	19.560392	19.197260

Tab. 4.1: ρ changed with $d = 10$ and $h_l = 1$

			model$\#1$		model $\#59$	
d	ρ	h_l	rise time (T_R)	overshoot	rise time (T_R)	overshoot
10	0	1	3.217464	76.768513	35.609671	0.509675
10	0	4	10.159198	23.403202	41.699064	0.637995
10	0	7	17.480939	6.127580	48.523603	0.749980
10	0	10	24.874413	0.219567	55.668963	0.830926
10	0	13	32.305714	0.366357	62.792331	0.873307
10	0	16	40.178461	0.525486	69.674838	0.874151
10	0	19	48.385933	0.664337	76.244483	0.835467

Tab. 4.2: h_l changed with $d = 10$ and $\rho = 0$

4.2.3 Tuning Algorithms

By defining different distances $d = 2 \ldots 15$ and shifting them with different minimum horizons $h_l = 1 \ldots 15$, closed-loop data was collected for every model. The maximum values of d and h_l are based on time delay measurements done in [3] where the latest response was measured after about 15 time steps ($\cong 5$ minutes). Based on the collected closed-loop data, the combination of d and h_1 connected with the best closed-loop behavior was selected. The two algorithms introduced in this section will weight the importance of the rise time and the overshoot constraint differently when picking the optimal tuning parameters.

Tuning Algorithm (A)

The first tuning algorithm assumes that a short rise time is the most important constraint for the closed loop and that the overshoot constraint can be relaxed to achieve the desired rise time. This assumption is based on the methodology used by the anesthesiologist, where the patient is paralyzed fast with a large initial bolus of Rocuronium.

Therefore the following routine for selecting d and h_l was implemented:

- $\bullet$ The fastest response that did not violate the overshoot nor the rise time constraint was selected (small dots).
- $\circ$ If there was no combination of d and h_l found that did not violate the overshoot constraint while being fast enough, the allowed overshoot was doubled. The fastest response that did not violate the new constraints was selected (little circles).
- $\star$ If tuning for this relaxed constraints was not possible either, the parameter combination leading to the smallest overshoot while still satisfying the rise time constraints, was selected (stars).

Tuning Algorithm (B)

The second tuning algorithm puts more emphasis on a small overshoot more than on a fast rise time. This approach would minimize the saturation time of the patient. However, it might introduce a longer waiting time for surgery to begin. This leads to the following tuning algorithm:

- $\bullet$ The response with the smallest overshoot that did not violate the overshoot nor the rise time constraint was selected (small dots).
- $\circ$ If there was no combination of d and h_l found that did not violate the overshoot constraint while being fast enough, the allowed overshoot was doubled. The response with the smallest overshoot that did not violate the new constraints was selected (little circles).

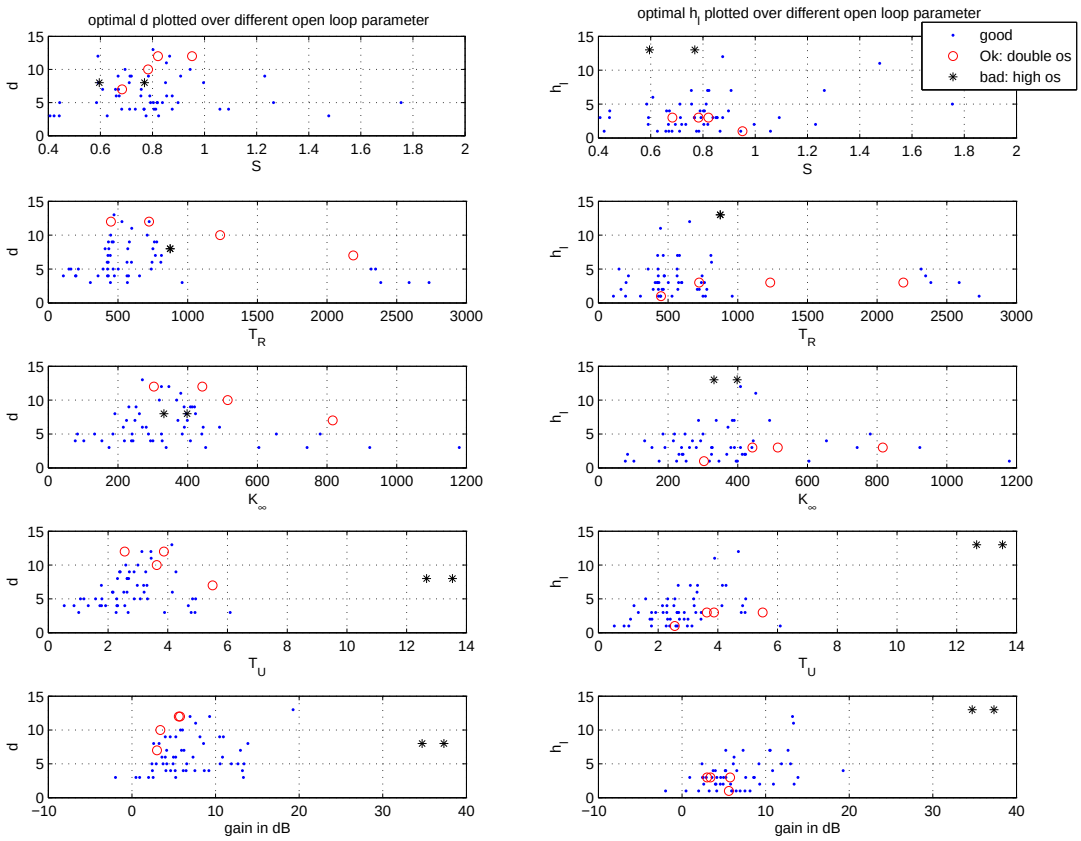

Fig. 4.5: (A): Chosen d and h_l over different parameters

⋆ If tuning for this relaxed constraints was not possible either, the parameter combination leading to the smallest overshoot while still satisfying the rise time constraints, was selected (stars).

4.2.4 Conclusions for Tuning

It can be seen in Fig. 4.5 and Fig. 4.6, that there is no good correlation between the optimal values for d and h_l and any open loop properties for tuning algorithm. Therefore the attempt to find a tuning routine based on open loop parameters was not successful. All schemes tested decreased the performance significantly. Therefore the optimal tuning parameters d and h_l calculated during the routines outlined above, will be used in the following simulations.

Fig. 4.5 - 4.8 show additionally that both tuning algorithms lead to controllers which are usually able to achieve good control performance (dots). Only the controllers for six models out of 59 can not be tuned to match the closed-loop requirements. Nevertheless four out of these six models ($\#25$, $\#47$, $\#52$, $\#59$) have

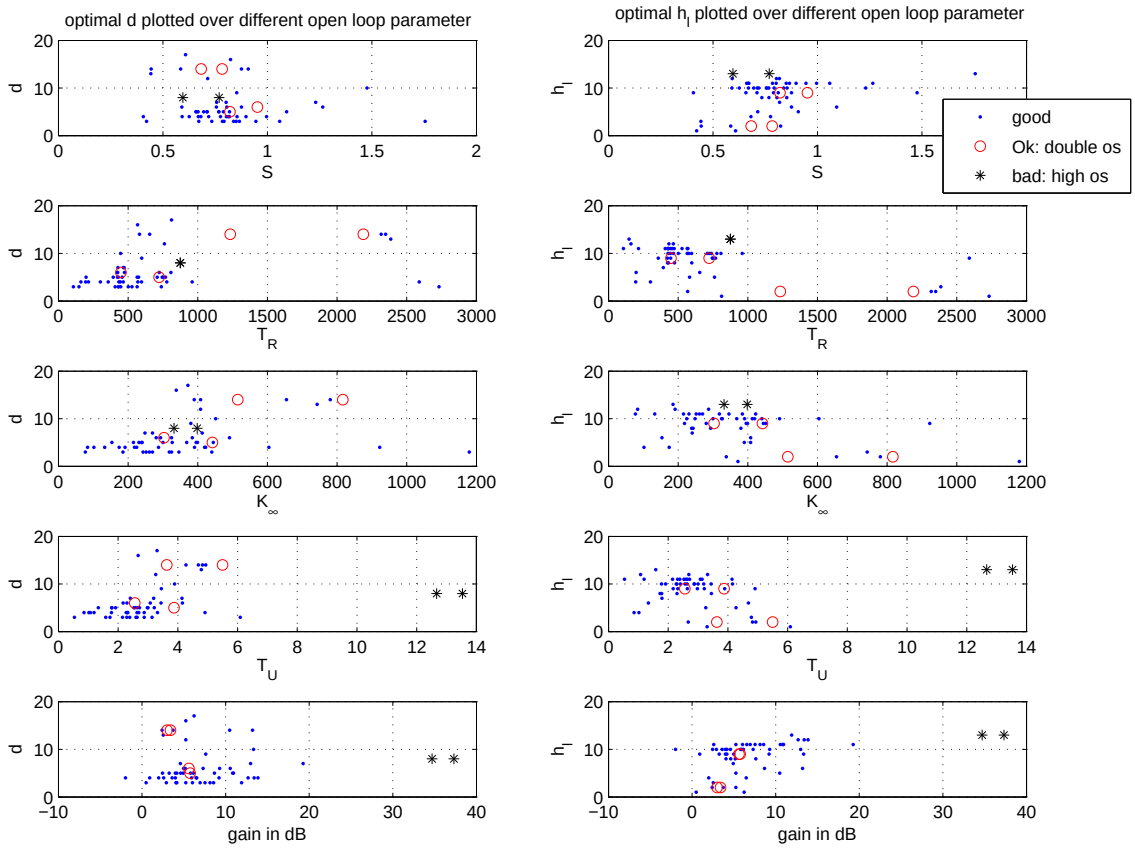

Fig. 4.6: (B): Chosen d and h_l over different parameters

only a slight increase in the overshoot while being still sufficiently fast (circles). Only two models ($\#26$, $\#36$) overshoot enormously when forced to meet the desired rise time (stars). Surprisingly this does not depend on the algorithm used.

Fig. 4.7 and Fig. 4.10 show, that model $\#26$ and $\#36$ can be separated by their start up time T_u as well as by their gain K. The red circle models ($\#25$, $\#47$, $\#52$, $\#59$) can not be separated as clearly, because they do not show significant differences in their step response nor bode plot. It is still believed, that models with a similar open loop start up time behave similar during control. Therefore the database of models was resorted according to the start up time T_u. The new model number will be indicated by $\#\#$ and a list of the corresponding model number $\#$ can be found in appendix A.

The closed-loop performance of the tuned controllers is illustrated in Fig. 4.11 - 4.16. The simulation was performed by using only the controller tuned for the selected model. Therefore, those graphs present the

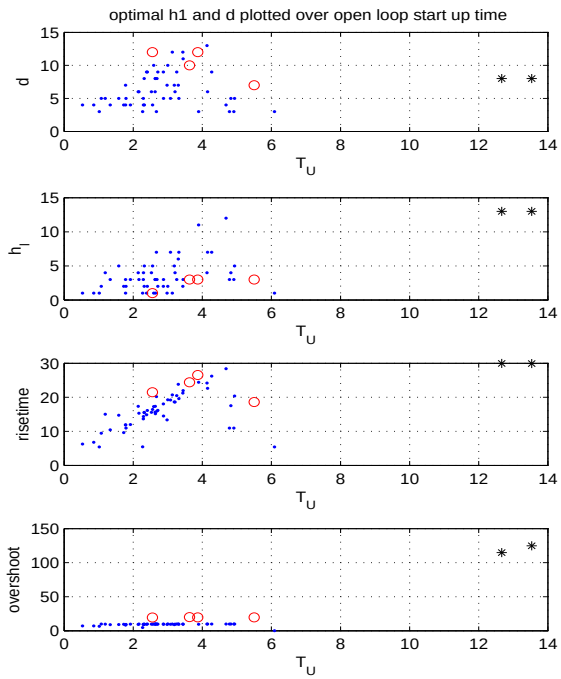

Fig. 4.7: (A): Closed loop performance over T_U

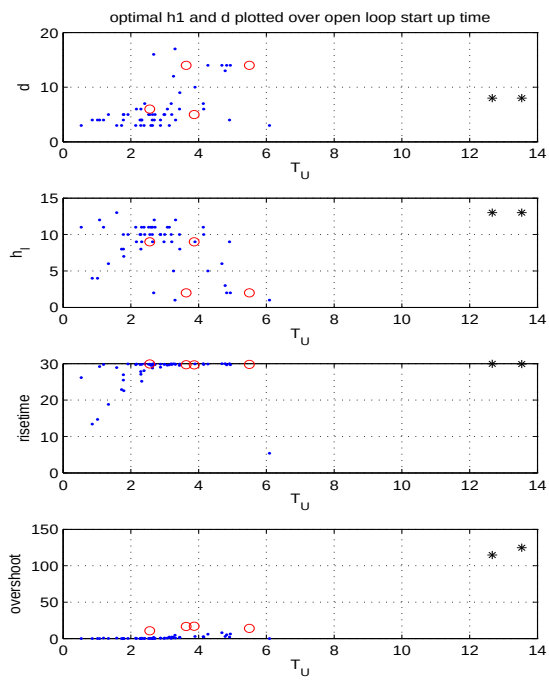

Fig. 4.8: (B): Closed loop performance over T_U

ideal case. This control performance could only be achieved in reality, if the patient model could be known a priori. In those figures the differences in the tuning algorithms can be seen clearly.

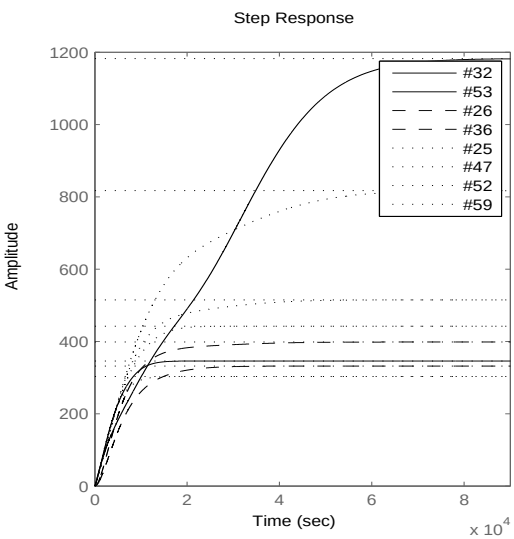

Fig. 4.9: Step response for selected models

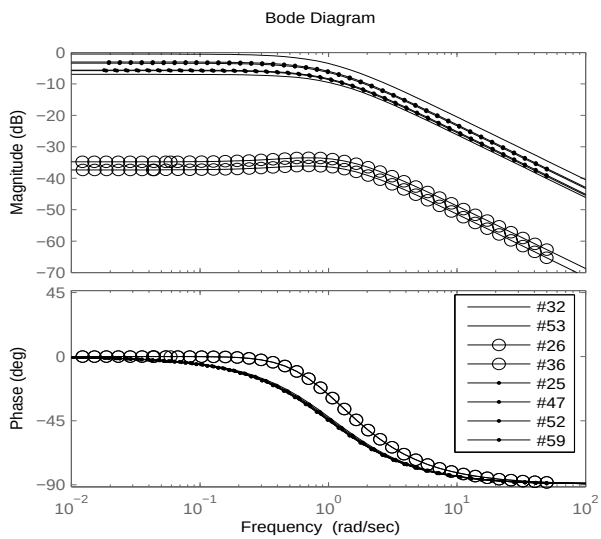

Fig. 4.10: Bode plot for selected models

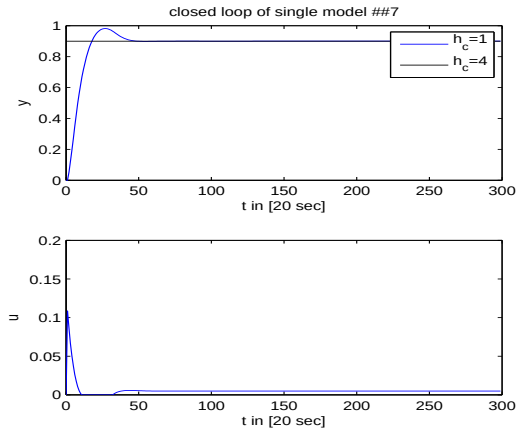

Fig. 4.11: (A): Model $\#\# = 7$: Good performance (dots)

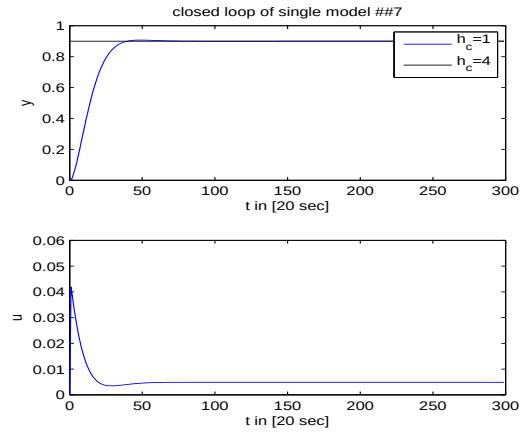

Fig. 4.12: (B): Model $\#\# = 7$: Good performance (dots)

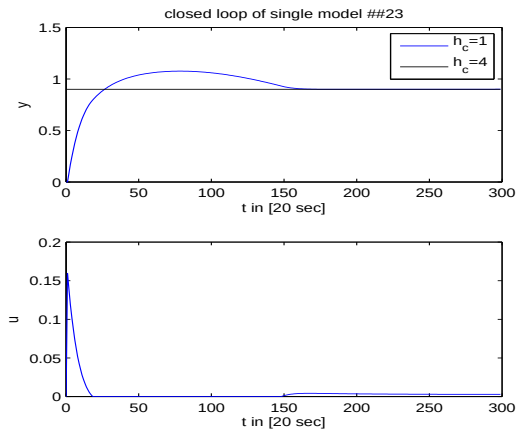

Fig. 4.13: (A): Model $\#\# = 23$: Double overshoot (circles)

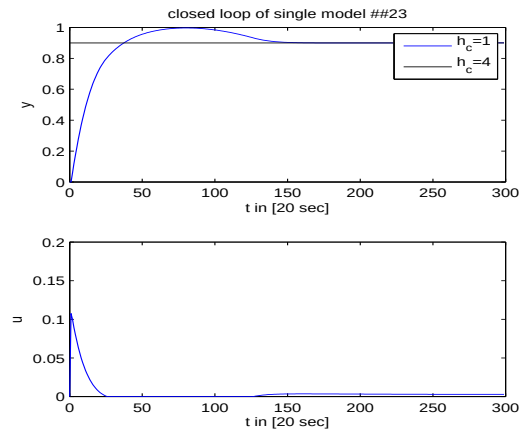

Fig. 4.14: (B): Model $\#\# = 23$: Double overshoot (circles)

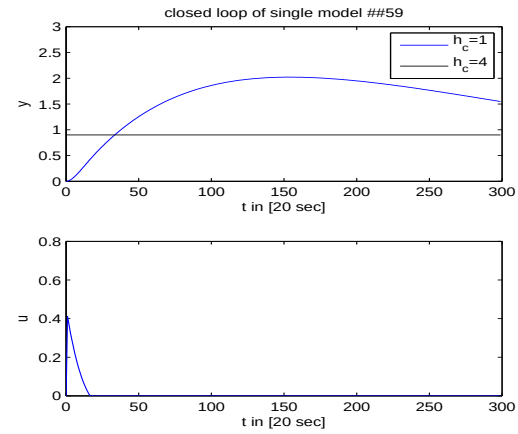

Fig. 4.15: (A): Model $\#\# = 59$: Bad performance (stars)

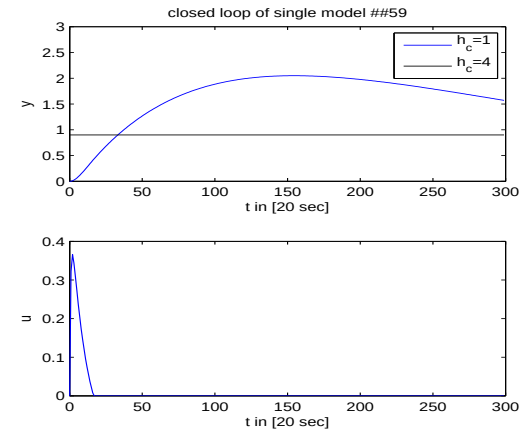

Fig. 4.16: (B): Model $\#\# = 59$: Bad performance (stars)

5 Closed Loop Simulations

In chapter 3 and chapter 4 the hybrid observation algorithm and the switched controller were introduced. It was argued that the B-MMAE and the S-RM-MMAE algorithm works best for the muscle relaxation problem. Furthermore two different tuning algorithms (A) and (B) of the parameters d and h_l for the GPC were explained. The closed-loop simulations in this section will compare the two hybrid observation algorithms B-MMAE and S-RM-MMAE and the two tuning algorithms (A) and (B). The aim is to investigate which algorithm works best and how applicable the introduced algorithms are to a closed-loop control of muscle relaxation in anesthesia.

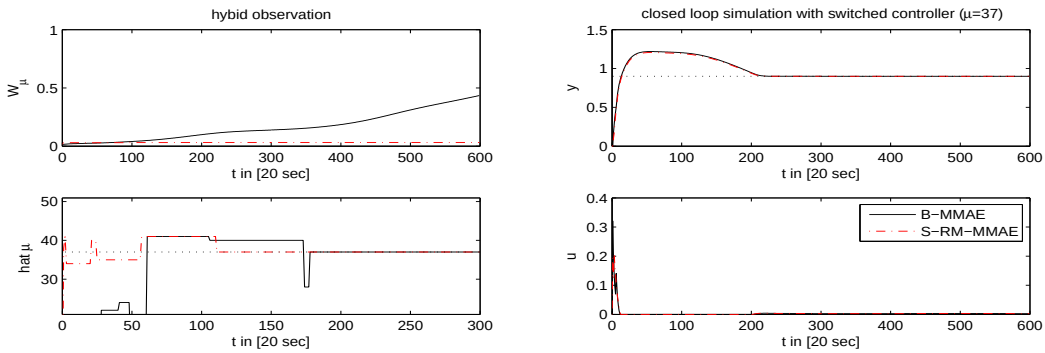

Fig. 5.1: (A): Closed loop simulation model ($\mu = \#\#37$) which showed good performance (dots)

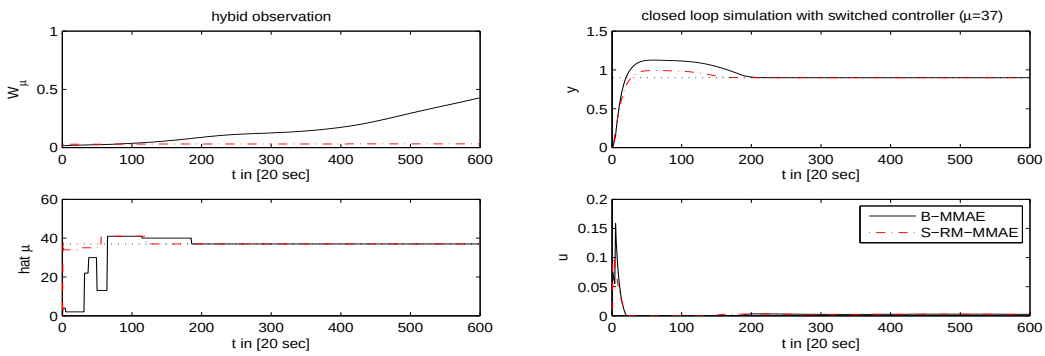

Fig. 5.2: (B): Closed loop simulation for model ($\mu = \#\#37$) which showed good performance (dots)

In section 4.2.4 was shown that it is not possible to find suitable tuning parameter for every model. Therefore representatives of every performance group were selected for closed-loop simulations:

- ● good performance (dots): $\#\#1$, $\#\#7$, $\#\#37$
- ○ slightly larger overshoot (circles): $\#\#23$, $\#\#46$, $\#\#56$
- ⋆ abnormal behavior (stars): $\#\#58$, $\#\#59$

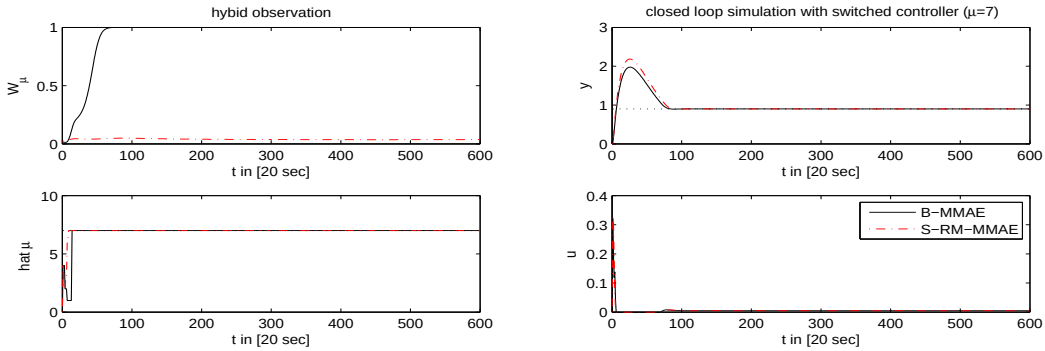

Fig. 5.3: (A): Closed loop simulation for model ($\mu = \#\#7$) which showed good performance (dots)

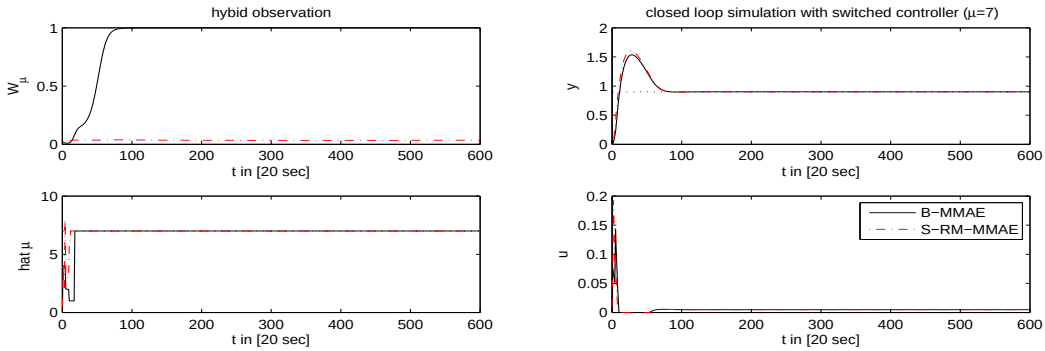

Fig. 5.4: (B): Closed loop simulation for model ($\mu = \#\#7$)which showed good performance (dots)

The main goal of this work was to design a closed loop which meets the following constraints:

overshoot $\leq 10\%$

rise time ≤ 10 minutes

To investigate how well the different algorithms meet these values, the four algorithms were combined with each other and closed-loop data was collected for every model. Tab. 5.1 shows the results for the eight selected models. The data for all models can be found in the appendix B.

The overshoot O_s and rise time T_r listed in Tab. 5.1 are illustrated in Fig. 5.1 - 5.8 which show the evolution of the input u and the output y on the right hand side. The performance of the hybrid observer is shown on the left hand side. The upper left graph displays the likelihood W_μ of the true model μ and the lower left graph displays the discrete state estimate $\hat{\mu}$.

By looking at those figures one can see, that the stochastic algorithm usually converges faster to the true model than the basic one. That was already shown in the open loop simulations. This means that the

	Tuning Algorithm (A)				Tuning Algorithm (B)			
	B-MMAE		S-RM-MMAE		B-MMAE		S-RM-MMAE	
$\#\#$	T_R in [min]	O_s in [%]	T_R in [min]	O_s in [%]	T_R in [min]	O_s in [%]	T_R in [min]	O_s in [%]
1	1.43	29.93	1.18	45.81	2.97	18.39	2.89	19.32
7	1.63	119.70	1.39	142.83	2.98	76.13	2.85	77.81
37	3.57	35.39	3.83	34.01	5.46	36.14	6.62	22.21
46	4.98	35.36	7.49	25.37	5.71	38.88	9.27	31.59
56	4.14	24.29	3.54	25.61	5.58	32.24	5.55	30.73
58	8.43	152.15	8.03	156.15	10.56	137.14	9.07	147.46
59	10.32	122.93	9.97	124.73	11.28	120.19	10.54	120.33

Tab. 5.1: Closed loop data

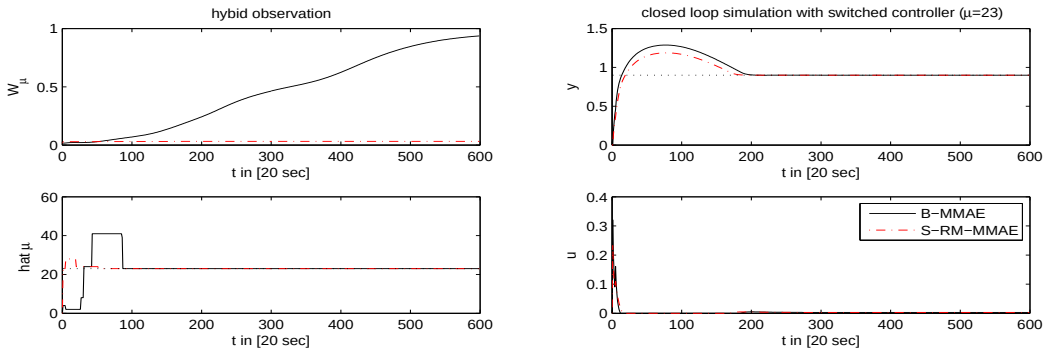

Fig. 5.5: (A): Closed loop simulation for model ($\mu = \#\#23$) that showed normal performance (cicles)

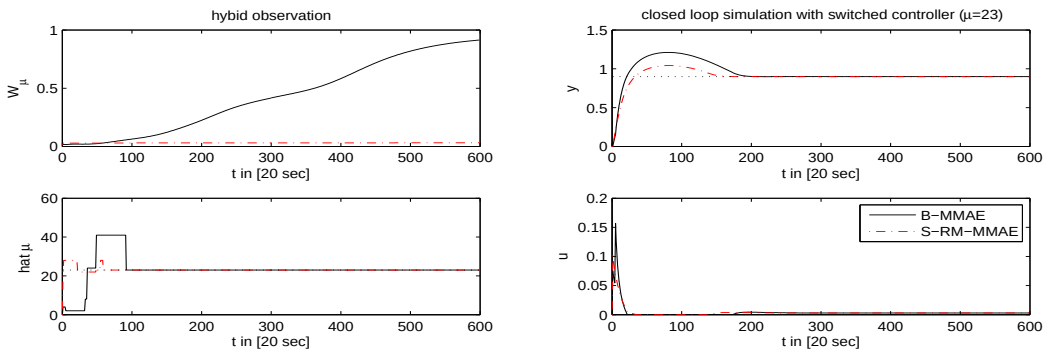

Fig. 5.6: (B): Closed loop simulation for model ($\mu = \#\#23$) that showed normal performance (cicles)

closed loop that uses the S-RM-MMAE observation algorithm switches to the right controller earlier than the closed loop using the B-MMAE algorithm. This leads to an improved closed-loop performance in many simulations. In the case of unexpected large overshoots as seen with model $\#\#7$ in Fig. 5.3, the use of the tuning algorithm (B) decreases the overshoot significantly.

Both Tab. 5.1 and Fig. 5.1- 5.8 show that the simulated closed loop is very fast, but non of the algorithms is able to meet the overshoot constraint. The reason for that is that all control action takes place in the first 10

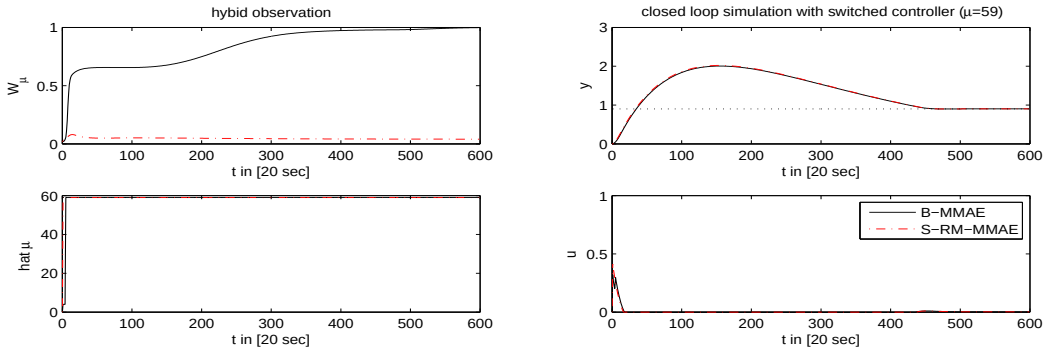

Fig. 5.7: (A): Closed loop simulation for model ($\mu = \#\#59$) that showed bad performance (stars)

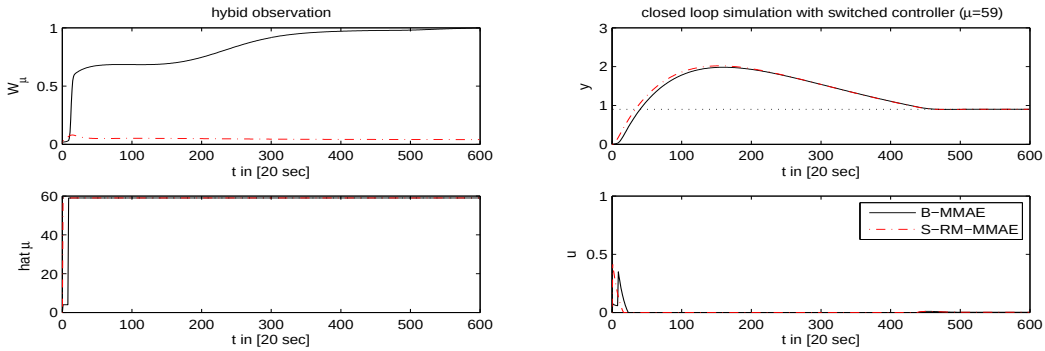

Fig. 5.8: (B): Closed loop simulation for model ($\mu = \#\#59$) that showed bad performance (stars)

samples ($\cong 4$ minutes) as can be seen in the lower right graph of Fig. 5.1- 5.8. This is a very short time for the hybrid observer to select the right controller. Therefore the largest control actions are performed, when the right controller has not been selected yet. This could lead to serious problems if the database would contain more models than the one used for these simulations.

It was nevertheless observed that the use of the S-RM-MMAE in combination with the tuning algorithm (B) can lead to a sufficiently good control performance for the normal models and the once with a slight overshoot. It is therefore concluded that an S-RM-MMAE observer and the tuning algorithm (B) are best suited for controlling muscle relaxation, based on the provided simulation results.

6 Conclusion

It was investigated in this work how well hybrid observation techniques in combination with a switched Generalized Predictive Controller are suited for a closed-loop control of muscle relaxation. Six different hybrid observers, all based on the Multiple Model Adaptive Estimation (MMAE) algorithm, were introduced and tested with step responses of the open loop system. Based on those simulations the Basic MMAE (B-MMAE) and the Stochastic MMAE that uses the mean value of the residuals (S-RM-MMAE) were selected to perform best for the given models. It was shown during the tuning of the different GPC that no good correlation between the optimal values of the tuning parameters and any open loop property could be found. Therefore the attempt to find a tuning routine for the GPC based on open loop parameters was not successful. The tuning was therefore done by applying two different optimization routines (A) and (B) to the collected closed-loop data to select the best tuning parameters. Finally, the observation algorithms B-MMAE and S-RM-MMAE were combined with the tuned switched controller and closed-loop simulations were performed. It was shown that the S-RM-MMAE in combination with the tuning algorithm (B) leads to the best closed-loop performance during simulation for the majority of the models.

It was additionally shown, that none of the tested algorithms was able to meet the overshoot constraint of the closed-loop system. It is believed that this problem arises because of the complexity of the investigated system. To achieve good control performance, the GPC has to act very aggressively on the system to make it sufficiently fast. This leads to large control actions during the first 10 time steps. During this time, the hybrid observer has not converged yet. Therefore the wrong controller is used to calculate the infusion rate of Rocuronium leading to a non optimal control input. When the discrete state estimate has converged to the right model after about 15 time steps the control inputs has no impact on the closed-loop performance any more. This problem could be addressed by including more information about the patient into the algorithm. It was shown, that the discrete state estimate converges faster if the number of models considered decreases. This could be done by using subgroups based on patient information (e.g. gender, race or weight).

All simulations done in this work used a randomly selected patient model of the database to generate an output y. This data is treated as the actual measurement by the algorithms. This leads to a very predictable system behavior that does not include disturbances, model uncertainties or measurement noise. Therefore more work needs to be done to ensure that the introduced algorithms perform equally well in actual surgery environments.

A Renumbering of Models

In the following table can be seen, which old model number $\#$ is connected to the model number $\#\#$ generated by renumbering the data base according to the start up time T_u of the models.

$\#$	$\#\#$		$\#$	$\#\#$
1	4		31	29
2	1		32	37
3	2		33	41
4	5		34	35
5	6		35	44
6	3		36	58
7	7		37	36
8	30		38	28
9	22		39	26
10	19		40	50
11	9		41	34
12	31		42	51
13	11		43	40
14	10		44	32
15	8		45	21
16	38		46	18
17	20		47	46
18	27		48	39
19	48		49	33
20	13		50	47
21	14		51	53
22	42		52	45
23	17		53	57
24	24		54	49
25	23		55	52
26	59		56	54
27	25		57	55
28	16		58	15
29	43		59	56
30	12			

B Closed loop data

In the following tables is shown, how the observation techniques B-MMAE and S-RM-MMAE and the tuning algorithms (A) and (B) influence the closed loop performance.

	Tuning Algorithm (A)				Tuning Algorithm (B)			
	B-MMAE		S-RM-MMAE		B-MMAE		S-RM-MMAE	
$\#\#$	T_R in [min]	O_s in [%]	T_R in [min]	O_s in [%]	T_R in [min]	O_s in [%]	T_R in [min]	O_s in [%]
1	1.43	29.93	1.18	45.81	2.97	18.39	2.89	19.32
2	2.50	6.25	2.40	6.37	4.21	15.54	4.18	15.49
3	2.91	1.87	1.91	6.37	4.24	18.08	3.99	18.04
4	2.61	26.34	2.36	30.58	4.14	19.17	3.70	21.10
5	2.98	31.70	2.75	31.69	5.35	29.71	4.23	31.11
6	2.10	37.04	1.82	42.34	3.79	29.83	3.56	30.35
7	1.63	119.70	1.39	142.83	2.98	76.13	2.85	77.81
8	2.99	11.26	3.21	9.61	5.42	20.99	5.60	17.05
9	2.58	25.85	2.69	24.19	4.12	23.90	5.57	21.02
10	2.94	16.41	3.24	14.66	5.40	22.53	5.85	16.23
11	2.68	18.57	2.89	17.02	4.24	21.59	5.60	16.14
12	2.33	22.25	2.74	19.88	3.98	26.64	5.52	18.10
13	3.13	29.33	3.39	26.65	5.41	28.29	5.81	18.52
14	3.06	22.89	4.16	14.63	5.43	28.15	7.55	19.75
15	2.55	1.99	1.89	4.65	3.93	27.88	7.59	17.06
16	2.91	20.80	4.37	10.43	5.39	24.62	6.60	18.41
17	3.18	20.93	4.85	9.10	5.43	24.71	7.62	17.76
18	1.81	50.26	2.63	39.43	3.69	42.39	5.50	16.83
19	3.65	26.89	4.79	13.89	5.47	27.37	7.64	16.64
20	3.12	32.81	4.19	18.45	5.34	29.90	6.61	15.96
21	1.66	81.00	2.48	53.37	3.48	57.60	4.14	25.95
22	3.73	30.06	4.58	16.41	5.51	29.68	7.63	13.87
23	3.65	43.15	5.12	32.01	5.50	46.15	9.23	28.16
24	3.35	27.64	4.64	17.00	5.44	27.53	7.62	17.21
25	2.95	34.23	4.10	22.76	5.37	31.14	6.63	20.30

	Tuning Algorithm (A)				Tuning Algorithm (B)			
	B-MMAE		S-RM-MMAE		B-MMAE		S-RM-MMAE	
$\#\#$	T_R in [min]	Os in [%]	T_R in [min]	Os in [%]	T_R in [min]	Os in [%]	T_R in [min]	Os in [%]
26	3.04	21.95	4.42	13.73	5.41	26.09	7.60	17.84
27	3.54	32.84	4.73	19.70	5.47	31.25	7.65	20.36
28	3.31	17.60	4.43	12.63	5.46	25.75	7.62	16.39
29	3.22	33.73	3.48	31.87	5.43	33.07	6.60	18.87
30	4.52	15.70	3.96	35.96	6.55	25.96	6.56	15.96
31	3.73	32.01	3.65	36.39	5.56	29.61	5.97	16.03
32	2.81	32.70	3.43	27.41	4.28	29.87	5.63	16.27
33	3.51	14.55	4.78	9.66	5.49	26.53	7.61	17.19
34	3.78	12.73	4.48	9.30	5.55	24.04	9.24	16.33
35	3.11	37.08	3.36	35.14	5.40	32.24	5.88	18.41
36	2.11	106.01	2.30	105.33	3.61	78.19	3.96	52.09
37	3.57	35.39	3.83	34.01	5.46	36.14	6.62	22.21
38	4.33	26.07	4.24	42.42	5.96	32.01	5.97	18.18
39	3.53	20.99	4.13	18.18	5.50	30.41	7.60	19.58
40	3.47	29.80	5.94	15.06	5.49	32.28	6.63	17.92
41	3.90	35.24	6.22	19.69	5.56	36.29	7.56	20.35
42	3.39	54.48	4.44	38.76	5.39	45.58	5.48	28.80
43	3.73	46.41	5.39	26.55	5.55	38.08	5.87	19.87
44	3.27	49.88	5.02	29.43	5.37	40.87	5.58	23.62
45	4.09	29.94	8.10	19.99	5.60	36.06	9.24	31.16
46	4.98	35.36	7.49	25.37	5.71	38.88	9.27	31.59
47	2.32	137.54	2.82	142.41	3.86	101.24	3.90	82.68
48	4.04	55.39	4.59	71.05	5.92	40.35	5.53	52.01
49	3.16	56.57	3.50	56.62	5.33	45.69	5.46	34.85
50	4.45	46.27	4.52	59.70	5.78	48.62	5.57	41.56
51	3.80	76.11	4.16	93.33	5.64	59.20	5.43	65.19
52	4.11	10.37	3.92	9.90	5.57	21.67	7.62	18.84
53	4.94	10.44	5.95	9.79	6.64	21.20	7.59	20.78
54	3.42	11.78	2.10	14.81	5.46	27.73	5.53	18.87
55	4.02	16.26	3.36	15.03	5.55	32.10	5.55	22.96
56	4.14	24.29	3.54	25.61	5.58	32.24	5.55	30.73
57	3.66	2.13	1.79	0.18	5.49	17.73	5.56	14.96
58	8.43	152.15	8.03	156.15	10.56	137.14	9.07	147.46
59	10.32	122.93	9.97	124.73	11.28	120.19	10.54	120.33

Bibliography

[1] S. Fekri, M. Athans, A. Pascoal *"Issues, progress and new results in robust adaptive control"*, Int. J. Adapt. Control Signal Process. 2006; 20:519-579

[2] I. Hwang, H. Balakrishnan, C. Tomlin *"State estimation for hybrid systems: applications to aircraft tracking"*, IEE Proc. -Control Theory Appl., Vol. 153, No. 5, September 2006

[3] T. J. Gilhuly *"Modelling and Control of Neuromuscular Blockade"*, Ph.D. Dissertation, University of British Columbia, Vancouver, Canada, July 2007

[4] A. D. MacLeod, A. J. Asbury, W. M. Gray and D. A. Linkens *"Automatic control of neuromuscular block with atracurium"*, British Jurnal of Anaesthesia, 63: 31-35, 1989

[5] E. Gepts, *"Pharmacokinetic concepts for TCI anaesthesia"* Anaesthesia, 53 (Supplement 1): 4-12, 1998

[6] C. C. Zervos *"Adaptive Control Based on Orthonormal Series Representation"*, Ph.D. Thesis, University of British Columbia, 1988

[7] G. A. Dumont, A. Elnaggar, A. Elshafei *"Adaptive Predictive Control of systems with time-varying time delay"*, Int. J. Adapt. Control Signal Process. 1993; Vol. 7: 91-101

[8] A. -L. Elshafei, G. A. Dumont, A. Elnaggar *"'Adaptive GPC Based on Laguerre-filters Modelling'"*, Automatica 1994, Vol. 30, No. 12, 1913-1920

[9] G. C. Goodwin, S. F. Graebe, M. E. Salgado *"Control System Design"*, Prentice Hall, 2001

[10] K. J. Åström, B. Wittenmark *"Adaptive Control"*, Addison-Wesley Publishing Company, 1995