Norwegian University of Science and Technology

TKP4580 - Chemical Process Technology, Specialization project

Scientific Machine Learning: Tuning out the Noise

Real-Time Optimization via Modifier Adaptation using Gaussian Processes

Author
Frida Bjørnstad Konow

Supervisor
Johannes Jäschke
Co-Supervisor
Evren Turan

NTNU

December 17, 2021

Abstract

Real-time optimization ensures process plant operation is continuously optimized to the economic optimum. In chemical process plants, developing an accurate process model to use in real-time optimization can be challenging. When doing static real-time optimization, utilizing measurements and plant gradients makes it possible to handle structural plant-model mismatch. Modifier adaptation is a type of real-time optimization that uses measurements in correction terms. The correction terms enables convergence to the plant optimum even with structural plant-model mismatch. However, first order modifiers depend on approximating plant gradients. Plant gradient estimation is a common challenge in control, especially in presence of measurement noise.

In this project thesis a proposed implementation of real-time optimization through modifier adaptation is presented, where the non-parametric regression approach Gaussian processes are used describe the plant-model mismatch. The objective is to enable real-time optimization with structural plant-model mismatch and in the presence of measurement noise. An implementation of standard modifier adaptation scheme is compared with a the proposed modifier adaptation with Gaussian processes scheme. The Williams-Otto reactor is used as case study for implementation of the different schemes.

Contents

1 Introduction **1**

2 Theory **2**
2.1 Real-Time Optimization . . . 2
2.1.1 Building Blocks . . . 3
2.1.2 Implementation . . . 3
2.1.3 Challenges . . . 4
2.2 Modifier Adaptation . . . 5
2.3 Gaussian Processes . . . 5
2.3.1 Linear Regression . . . 6
2.3.2 Gaussian Process Regression . . . 6
2.3.3 Kernel Functions . . . 7
2.3.4 Hyperparameters . . . 8

3 Problem Formulation **9**
3.1 Steady State Optimization Problem . . . 9
3.2 Necessary Conditions of Optimality . . . 9
3.3 Modifier Adaptation . . . 10
3.4 Modifier Adaptation with Gaussian Processes . . . 12

4 Case Study: Williams-Otto Reactor **14**
4.1 Plant and Model Equations . . . 15
4.2 Optimization Problem . . . 16

5 Results and Discussion **17**
5.1 The Case Studies . . . 17
5.2 Programming Environment . . . 18
5.3 Standard Modifier Adaptation without Noise . . . 18
5.4 Standard Modifier Adaptation with Noise . . . 20
5.5 Modifier Adaptation with Gaussian Processes without Noise . . . 22
5.6 Modifier Adaptation with Gaussian Processes with Noise . . . 23
5.6.1 Base case . . . 23
5.6.2 GP Version 2: Additional Training Data at New Point . . . 25
5.6.3 GP Version 3: Two iid Estimates of Plant Response at $\mathbf{u}_k$ . . . 26
5.7 Possible Improvements . . . 27

6 Conclusion **30**

A Gradient Approximation - Finite Differences **33**

B Algorithms **34**
B.1 Standard Modifier Adaptation . . . 34
B.2 Modifier Adaptation with Gaussian Processes . . . 35

List of Figures

2.1 Levels of process control hiearchy and belonging time scales. Figure reproduced from Figure 19.1 in [1]. 2
2.2 Block diagram for static RTO. Figure reconstructed from figure in [2]. 4
2.3 Block diagram of the modifier adaptation scheme. 5
2.4 Three random samples from a GP prior distribution and three random samples from the GP posterior with the belonging GP mean, a 95% confidence interval and the actual function. Figure created based on Figure 2.2 in [3]. 7
3.1 Graphical representation of the modified constraint function $G_{m,i,k}$ with the belonging modifiers $\epsilon_{i,k}$, λ_k^{Φ} and $\boldsymbol{\lambda}_k^{G_i}$. Figure based on Figure 1 in [4] and Figure 1 in [5]. 11
4.1 Illustration of the Williams-Otto reactor with feed streams F_A and F_B, outlet stream F and the plant reactions. 14
5.1 Plot of input iterations for standard MA without noise. Each star represents one iteration and the dotted line shows the order of the iterations. The final iteration is labelled u_k^f. g_1 and g_2 are the constraints that bound the feasible region indicated with the blue area. 19
5.2 Plant profit, T_R and F_B plotted against iterations for standard MA without noise. 19
5.3 Weight fraction x_A and x_G plotted against iterations for standard MA without noise. The constraint values are indicated with red dotted lines. 20
5.4 Plot of standard MA with 4 different noise levels. Plot A and B have equal noise for constraint and cost function. Plot C and D have cost noise calculated from Equation 5.1. 20
5.5 Plot of input iterations for standard MA with different filter parameters. k_1 and k_2 are filter parameters for the inputs, a, b and c are filter parameters for the modifiers. The measurement noise level is equal to $2 \cdot 10^{-7}$ for all plots. 21
5.6 Plot of input iterations for MA with GP without noise. 22
5.7 Plot of input iterations for the base case of MA with GP for 4 different noise levels. 24
5.8 Plot of input iterations for the second version of MA with GP for 4 different noise levels. 25
5.9 Plot of input iterations for the third version of MA with GP for 4 different noise levels. 26

List of Tables

4.1 Kinetic parameters for the plant and model reactions. Values from the plant are obtained from [6] and model parameters were decided through personal communication. 16
4.2 Prices for the products P and E, and the feed reactants A and B [7]. 16
5.1 The optimal values of the flow rate of B, F_B^*, the reactor temperature, T_R^*, the profit or negative cost, $-\Phi_p^*$, and the constrained weight fractions, x_A^* and x_G^*, from simulation of the standard MA without noise. 18
5.2 The optimal values F_B^*, T_R^*, $-\Phi_p^*$, and x_A^* and x_G^*, from simulation of MA with GP without noise. 22
5.3 Final hyperparameters, RBF length scale, l, and white noise, σ_n^2, for the three GPs for the base case MA with GP. 25
5.4 Final hyperparameters, RBF length scale, l, and white noise, σ_n^2, for the three GPs for the second version of MA with GP. 26
5.5 Final hyperparameters, RBF length scale, l, and white noise, σ_n^2, for the three GPs for the third version of MA with GP. 27

Nomenclature

Acronymes

Abbreviation	Description
RTO	Real-time optimization
MA	Modifier adaptation
GP	Gaussian processes
NLP	Nonlinear programming
NCO	Necessary conditions of optimality
LICQ	Linear independence constraint qualification
KKT	Karush–Kuhn–Tucker (conditions)
IID	Independent and identically distributed
MPC	Model predictive control
RBF	Radial basis function

Symbol list

Symbol	Description	Unit
Theory		
Φ	Cost function	\$/s
x	State	-
u	Input	-
d	Disturbance	-
F	Feed flow rate	kg/s
P	Product flow rate	kg/s
Q	Energy usage	energy/s
p_F	Feed price	\$/kg
p_F	Product price	\$/kg
p_Q	Energy price	\$/energy
$\hat{d}$	Estimated disturbances/parameters	-
y	Measurement/output	-
u_k^*	Optimal input calculated from RTO	-
u_{lower}	Input for lower level	-
f	Unknown function	-
ν	Noise in data	-
α	Regression parameter	-
β	Regression parameter	-
$\mathcal{N}$	Normal distribution	-
σ^2	Variance	-
$\mathbf{u}$	Input vector	-
μ	Mean function	-
$k(\cdot,\cdot)$	Covariance/kernel function	-
$\overline{\mathbf{u}}$	Input data	-
$\overline{\mathbf{y}}$	Output data	-
y_*	Predicted output	-
k_{RBF}	Radial basis function/squared exponential kernel	-
σ_f^2	Radial basis function variance	-
l	Radial basis function length scale	-
k_w	White noise kernel	-
σ_n^2	White noise variance	-
K	Combined RBF and white noise kernel	-
LML	Log-marginal-likelihood function	-
θ	Collection of hyperparameters	-

Symbol	Description	Unit
Problem formulation		
Φ_p	Plant evaluated cost function	\$/s
Φ	Model evaluated cost function	\$/s
$\mathbf{y}_p$	Measured outputs	-
$\mathbf{y}$	Estimated outputs	-
$G_{p,i}$	Plant evaluated constraints	-
G_i	Model evaluated constraints	-
n_g	Number of constraints	-
n_u	Number of inputs	-
n_y	Number of outputs	-
U	Upper and lower input bounds	-
$\mathbf{u}^{\mathrm{L}}$	Lower input bound	-
$\mathbf{u}^{\mathrm{U}}$	Upper input bound	-
$\mathbf{F}$	Process model	-
$\mathcal{A}$	Set of active constraints	-
$\mathbf{u}^*$	Optimal input	-
$\mathcal{L}$	Lagrangian function	-
$\boldsymbol{\gamma}$	Lagrangian multiplier	-
k	Current iteration	-
$\mathbf{u}_k$	Current operating point	-
$\Phi_{m,k}$	Modified cost function at operating point	-
$G_{m,i,k}$	Modified constraint function i at operating point	-
$\epsilon_{i,k}$	Zero order modifier	-
$\boldsymbol{\lambda}_k^{\Phi}$	First order modifier for cost function	-
$\boldsymbol{\lambda}_k^{G_i}$	First order modifier for constraint function	-
$\mathbf{K}$	Input filter matrix	-
k_i	Input filter parameter for input i	-
a_i	Zero order modifier filter parameter for constraint i	-
b_i	First order modifier filter parameter for constraint i	-
c	First order modifier filter parameter for cost function	-
y_*	Predicted output	-
$\epsilon_{i,k,GP}$	Zero order modifier calculated with GP	-
$\boldsymbol{\lambda}_k^{\Phi_{GP}}$	First order modifier for cost function calculated with GP	-
$\boldsymbol{\lambda}_k^{G_{i,GP}}$	First order modifier for constraint function i calculated with GP	-
$(GP)^{(\Phi_p - \Phi)}$	GP calculating plant-model difference in cost function	-
$(GP)^{(G_{p,i} - G_i)}$	GP calculating plant-model difference in constraint function i	-
Case study		
A	Reactant	-
B	Reactant	-
C	Intermediate product	-
P	Product	-
E	Product	-
G	Byproduct	-
F_A	Feed flow rate of A	kg/s
F_B	Feed flow rate of B	kg/s
F	Total flow rate	kg/s
x_i	Weight fraction of $i \in \{A, B, C, P, E, G\}$	-
k_i	Reaction rate constant for reaction $i \in \{1, 2, 3, 1^*, 2^*\}$	s^{-1}
A_i	Pre-exponential factor for reaction $i \in \{1, 2, 3, 1^*, 2^*\}$	s^{-1}
$E_{a,i}$	Activation energy for reaction $i \in \{1, 2, 3, 1^*, 2^*\}$	K
T_R	Reactor temperature	
r_i	Reaction rate for reaction $i \in \{1, 2, 3, 1^*, 2^*\}$	s^{-1}
M_t	Total mass	kg
P_i	Price of $i \in \{A, B, P, E\}$	\$
σ	Measurement noise	-
σ_{Φ}	Cost function noise	-

1 Introduction

This project examines the use of Gaussian processes in a modifier adaptation framework to enable real time optimization despite plant-model mismatch and measurement noise.

Real-time optimization (RTO) of process systems aims to ensure system operation, while meeting quality and safety constraints and optimize an economic objective. Optimization based solely on a model, will often not converge due to structural plant-model mismatch. Even with accurate models, optimality might be infeasible if the system is exposed to disturbances that changes the optimum. As a result, real time optimization needs process measurements in order to ensure convergence and optimal operation [7].

Modifier adaptation (MA) is a type of RTO that utilize measurements in the optimization with correction terms called modifiers. The modifiers represent the plant-model mismatch for cost and constraint functions. Compared to standard RTO, MA possess the ability to converges to the plant optimum, given certain prerequisites, even with plant-model mismatch. However, it requires estimation of the plant gradients. Accurate gradient approximation can be costly and sometimes unattainable, especially when the system exposed to measurement noise.

Gaussian process (GP) regression is a probabilistic, nonparametric, Bayesian approach to regression that is gaining attention in machine learning field [8]. It can be used to estimate unknown functions, and interpreted as an extension of multivariate normal distribution to infinitely many random variables [7]. GP regression has several advantages, working well on small datasets, having the ability to provide direct uncertainty measurements on the predictions and capture complex unknown functions. These abilities are highly sought in the setting of RTO.

The objective for the project is is to implement RTO through MA and use GP to represent the plant-model mismatch in presence of measurement noise. First a standard MA will be formulated, then extended by introducing GP. Three different version of implementing GP in the MA framework will be proposed. All schemes will be applied in a case study with the Williams-Otto reactor with and without measurement noise to compare the performance in terms of efficiency and convergence ability.

2 Theory

In this section theory and background relevant for the project will be introduced. We start by presenting RTO, with its building blocks and implementation, followed by some challenges with RTO. Further the concept of modifier adaptation is introduced. Gaussian processes are presented from a practical perspective to describe how it will be applied in the project.

2.1 Real-Time Optimization

There are five main levels in the process control hierarchy, presented in Figure 2.1. The hierarchy represents control decisions within optimization, monitoring and data acquisition in the different relative time scales of a plant [1]. Each block in the hierarchy is conceptual, as the different blocks might use the same computations. The highest level is called planning and scheduling and includes demand forecasting, supply chain management and scheduling of raw materials and product. This level has a long time scale that can vary from days to months and is typically carried out by the plant manager. The decisions involve planning of retailing, distribution, transportation and manufacturing, and the output is obtaining production and inventory targets and optimal operating conditions for the plant. The fourth level is real time optimization (RTO), both plantwide and of the individual units, and includes parameter estimation, supervisory control, and data reconciliation, with a time scale of hours to days. Decisions are made considering the explicit economical objectives. The RTO level can be interpreted as an online calculation of the optimal set-points for the lower level called supervisory control [1].

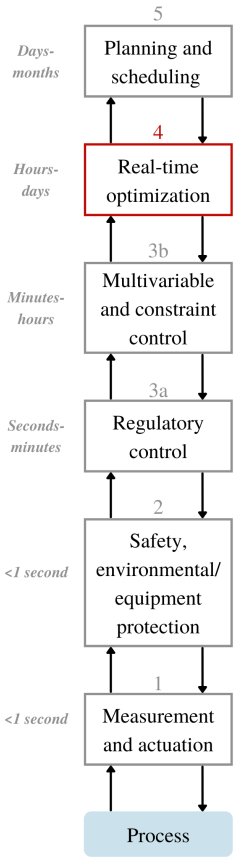

Figure 2.1: Levels of process control hiearchy and belonging time scales. Figure reproduced from Figure 19.1 in [1].

Level 3b, multivariable and constraint control or model predictive control (MPC), allows the plant to operate near the constraints. In level 3a regulatory control techniques are performed for single- or multi-loop control, to guarantee that variables are kept on the set points. Level 2 includes safety, environmental and equipment protection, while level 1 represents measurements and actuation. The arrows back and forth the levels symbolize communication between the sequential levels, where the higher sets the objective for the lower level. The lower-level pass on current operation data to the higher level, which it can base the decisions on. For this project level 4 with RTO will be the center of attention.

RTO optimizes the plant economics. From around the 1990s, RTO has gained attention when it comes to plant control [1]. The reason is that use of RTO can lead to considerable economic gain, since it continuously optimizes the operating conditions based on an economic objective or cost function [5]. If variables in the cost function change frequently, for example the price of electricity, RTO can adjust the optimal operating conditions for high and low electricity prices. RTO can smoothly be built into computers with control systems, because of rapid development in hardware and software field [1].

2.1.1 Building Blocks

In RTO there are three main building blocks; the economic model, the process model and the process constraints [2]. The steady state optimization problem is stated in Equation 2.1, where $\Phi(x, u, d)$ is the economic model, $f(x, u, d)$ is the process model and $g(x, u, d)$ is the process constraints:

$$\begin{aligned} \min_{u} \quad & \Phi(x, u, d) \\ \text{s.t.} \quad & f(x, u, d) = 0, \\ & g(x, u, d) \leq 0, \end{aligned} \tag{2.1}$$

The economic model, also called the cost function, $\Phi(x, u, d)$, is dependent on the states x, the inputs u and the disturbances d. The general objective for the process operation in the RTO is to minimize the cost and maximize the profit. It typically includes the costs of the raw material for the feed, value of the products and the cost of utilities, often energy. Operational costs can also be included, although they can often be assumed fixed for the time scale of interest [2]. With that assumption and energy as the only utility the cost function can be expressed as

$$\Phi = \sum p_F F + \sum p_Q Q - \sum p_p P, \tag{2.2}$$

where F, Q and P indicates feed flow rate, energy usage per time unit and product flow rate respectively [2]. p_i is the prices of the different flows and energy in unit [\$/kg] and [\$/energy].

$f(x, u, d)$ in Equation 2.1 is the process steady state model. This model is usually derived through fundamental equations, like mass and energy balances, or through experimental data. The role of the model is to use the operating conditions for each unit, as flow variables, temperatures, and pressures, to obtain the yield of product, and other variables needed in the cost function. The process constraints $g(x, u, d)$ are typically set from requirements on product purity, amount of waste and the equipment's capacity [9]. In practice, are often f and g merged in the problem formulation as a set of constraints, where f are equality constraints and g are inequality constraints.

2.1.2 Implementation

A block diagram for a conventional steady state RTO is presented in Figure 2.2 [2]. The figures show the steps needed for implementation of RTO. It consists of parameter estimation, including steady state detection, which yields the parameters needed for the "Static RTO", where the plant is re-optimized to find the new optimal steady state.

The measurements y are provided from the process. Due to steady state models in the optimization in RTO, it is necessary to ensure that the process is at steady state before executing the optimization. Therefore, there is a block that uses the measurements and continuously check if the process has reached steady state. This is the steady state detection. There are several methods that are possible to use for the purpose of detecting steady state. Statistical tests are one common group of methods [2].

In the parameter estimation, the objective is to estimate the unmeasured disturbances and the parameter values d, given the steady state process measurements y, inputs u and a process model. An optimization problem is solved, where the norm of the deviation between the measured y and calculated values for y based on the process model, is minimized. The constraints for the optimization problem are the process model and the operational constraints. When solving this optimization problem, the calculated $\hat{d}$ yields the smallest deviation between system measurements and model subject to the constraints [1].

When steady state is detected and the unknown parameters are calculated, the system is re-optimized in the static RTO block. The optimization problem in Equation 2.1 is solved, which results in new inputs that are implemented as set-points in the plant.

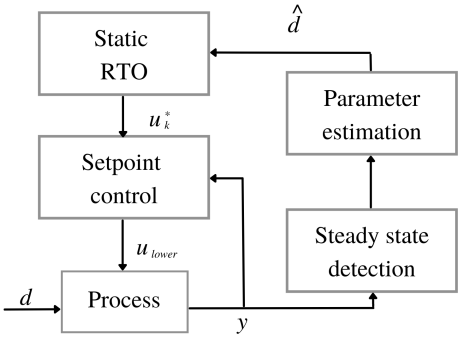

Figure 2.2: Block diagram for static RTO. Figure reconstructed from figure in [2].

2.1.3 Challenges

To generalize RTO can be seen as a two-step approach consisting of 1) parameter estimation, and steady state detection, and 2) optimization, which is repeatedly executed in iterations. This approach needs two requirements to be fulfilled to perform well. First there needs to be small structural plant-model mismatch and secondly that the difference in the optimal operating points between each iteration give enough excitation to enable estimation of the unknown model parameters. However, it is rare that both requirements are fulfilled in reality [4].

Whenever there is a mismatch between the model and the plant, the result is generally that the RTO is not able to satisfy the necessary conditions for optimality, NCO (Section 3.2). The reason is that the parameters estimated, will not be a sufficient representation of the plant resulting in calculated setpoints that are sub-optimal. To overcome this challenge, use of measurements and identification of plant gradients can be introduced to ensure that the NCO are fulfilled [5]. Further, a solution to ensuring enough excitation to estimate the model parameters, is to increase the number of identifiable parameters. To overcome these challenges a group of methods called fixed-model methods have evolved, where the model-parameter update is not necessary [10]. The choice of RTO method always depends on the plant that is controlled, but there are three main features that are highly emphasized. The first is guaranteed optimality upon convergence, secondly that the method converge quickly and finally that the convergence leads to a feasible point with no constraint violation [5].

2.2 Modifier Adaptation

Modifier adaptation (MA) is a fixed-model RTO method. Common for all fixed-model methods, is that they use a nominal process model together with plant measurements in the optimization [10]. A nonlinear programming, NLP, problem including a nominal process model is solved repeatedly [4]. In contrast with two-step RTO approach, the parameters are not estimated at each iteration. Instead, the measurements are used to update the cost and constraint functions. In MA measurements are utilized in correction terms called modifiers that are calculated prior optimization at each iteration. The modifiers ensure that NCO is fulfilled when the optimization converges [4]. In MA schemes, the modifiers are calculated based on the deviation between the plant measurements and the process model. For standard MA, two orders of modifiers are introduced: zero and first order. Zero order modifiers represent the direct deviations, while first order represent the deviations in the gradients with respect to the inputs. In Section 3.3 the mathematical equations for MA are presented. The main advantage of MA is its ability to converge to the optimum even in presence of structural plant-model mismatch [5].

The standard MA scheme is illustrated with a block diagram in Figure 2.3. Compared with the two-step RTO approach in Figure 2.2, the parameter estimation block is replaced with a block for modifier calculation. As a result, it is not the estimated parameters, but the modifiers that are fed to the optimization block. Here the optimizer calculates the optimal inputs based on the modified optimization problem with the modifiers present in the cost and the constraint functions. The rest of the scheme is unchanged, where the optimal inputs provide the setpoints for the lower level.

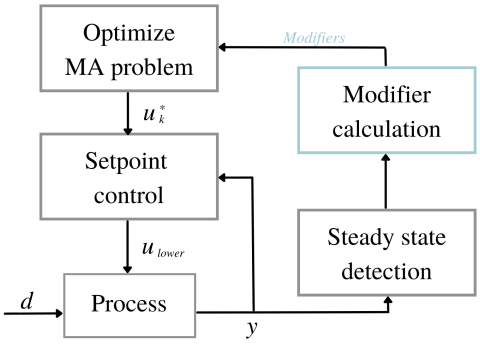

Figure 2.3: Block diagram of the modifier adaptation scheme.

In terms of fulfilling the three desired RTO features, the second property of quick convergence and third property on feasible convergence, both rely on the plant and process model. It can be observed that these properties might be contradictory, as fast convergence can mean large steps for each iteration, while feasible convergence favours smaller steps are desired [5]. MA fulfills the first property of converging to the plant optimum, given accurate plant evaluations and gradients. As mentioned MA has an advantage of fulfilling the NCO compared with two-step RTO approach, even with structural plant model mismatch. However, obtaining accurate gradients introduce a new challenge, as approximating sufficient plant gradients can be costly and sometimes infeasible. The challenge amplifies for systems with high measurement noise levels. Therefore, measures to handle gradient estimation in the presence of noise could be beneficial for MA implementation.

2.3 Gaussian Processes

Gaussian processes, or GP, provides a non-parametric approach to Bayesian regression, which has gained attention for its abilities in the machine learning [11], namely it's ability to perform well despite few data points and describe uncertainty. To introduce GP let us first consider linear regression.

2.3.1 Linear Regression

Regression is commonly used to study how one or several independent variables, u, affect a dependent variable, y. The independent variables can also be referred as inputs or explanatory variables, and the dependent variable as the output or response. The simplest form of regression is linear regression and can be written on the form

$$f(u) = \alpha + \beta u, \qquad y = f(u) + \nu. \tag{2.3}$$

$f(u)$ is the function value and y is the observed value [3]. ν explains the noise in the observed data, which is typically assumed to be independent, and identically Gaussian distributed with zero mean and variance σ^2 given as [3]

$$\nu \sim \mathcal{N}(0, \sigma^2). \tag{2.4}$$

In Equation 2.3 α and β are regression parameters. To obtain a function for the relationship between the variables $f(u)$, the parameters must be decided. The strategy for deciding the parameters in linear regression is to find the "best fit" for the data points, by minimizing sum cost function, typically the sum of squared errors. This means the parameters minimize the squared distance from each data point to the regression line. If the data points are not well described by a linear function, either increasing order of the polynomial or introducing are several input variables can be considered. In statistical practice, these two cases can be interpreted the same way, if one defines the higher order input, e.g. u^2, as a new independent variable [12].

When increasing the order of the regression polynomial or adding independent variables, several parameters will be included in the regression model. One challenge with regression can be if there is no obvious "best fit" to the data. Statistical parameters like R^2, p-values and mallows c_p can be studied to determine the best fit [12]. However, the best regression model based on one of these parameter might be contradictory to the best regression model based on another parameter. Therefore it can be difficult to determine which model to use. One way to overcome this challenge, is to use nonparametric regression like Gaussian process regression.

2.3.2 Gaussian Process Regression

Gaussian process (GP) regression is a nonparametric, probabilistic regression approach [11]. Regular regression yields *one* regression function and the uncertainty for the function, but in practice there could possibly be an infinite number of functions that describe the data. GP regression considers all these functions as it describes a distribution over functions [11].In addition, it possesses a useful property to quantify the uncertainty in a prediction. It is a Bayesian approach to regression with a prior and posterior distribution [3].

Gaussian processes, can be defined as follows:

Definition 2.1 (Gaussian Process)**.** *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution* [3]

When GP is used in the context of regression to predict an unknown function, it is referred to as GP regression. Further a brief introduction on how to utilize GP with respect to this project will follow. A full introduction and mathematical derivations can be found in [3].

Consider an unknown function $f : \mathbb{R}^{n_u} \mapsto \mathbb{R}$ that we want to approximate using GP. The function is dependent on the input $\mathbf{u}$ of dimension n_u and the output of the unknown function can be referred to as y, with the noise ν:

$$y = f(\mathbf{u}) + \nu. \tag{2.5}$$

Note that this is analogous to the relation presented in the last part of Equation 2.3, except now the function is unknown. A Gaussian process prior is defined by a mean function $\mu(\mathbf{u})$,

and a covariance or kernel function, $k(\mathbf{u}, \mathbf{u}')$ of $f(\mathbf{u})$ as stated in Equation 2.6. The mean function describes the average of all the functions included in the distribution. The kernel function can either be constructed or chosen from pre-defined kernels (Section 2.3.3). The GP can be interpreted as an infinite multivariate Gaussian distribution [7]. This implies that any collection of points described by a GP are joint Gaussian distributed as stated in Definition 2.1. The prior GP does not contain any information of observed data.

$$f(\mathbf{u}) \sim GP(\mu(\mathbf{u}), k(\mathbf{u}, \mathbf{u}')). \tag{2.6}$$

Assume that there are n_p observed input-output data points $(\mathbf{u}_i, y_i), i = 1, ..., n_p$. The input vector can be given as $\overline{\mathbf{u}} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{n_p}]$ of dimension $n_u \times n_p$ and output $\overline{\mathbf{y}} = [y_1, y_2, ..., y_{n_p}]^{\mathrm{T}}$ of dimension $n_p \times 1$. The datapoints are typically labeled the GPs "training data" [3]. The objective is to enable a sufficient prediction of a new output, y_*, based on the given data points. In Bayesian regression, this is done by conditioning the prior on the training data, to give a posterior distribution [3]. By construction, the posterior of the GP can be calculated with relative ease.

Figure 2.4 illustrate one example of three random prior distributions followed by three random posteriors restricted by the training data. The posteriors are plotted together with the mean function, μ, of all the posteriors and the actual function that the data was sampled from. The figure gives some intuition of how the update of the prior distribution is done by "rejecting" all functions that does not match the given data. For example, all the three prior functions displayed are rejected, as they do not go through the data points.

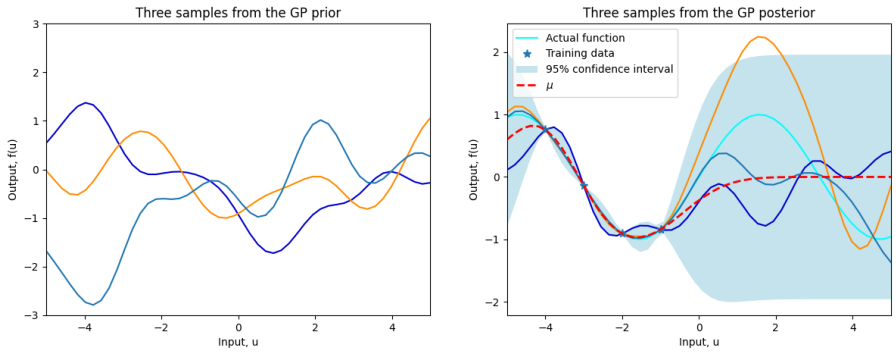

Figure 2.4: Three random samples from a GP prior distribution and three random samples from the GP posterior with the belonging GP mean, a 95% confidence interval and the actual function. Figure created based on Figure 2.2 in [3].

Given the unknown function f, a simplified notation that describes the output distribution y from the GP given the datapoints $(\overline{\mathbf{u}}, \overline{\mathbf{y}})$ can be written as the following,

$$y_* = (GP)^f(\mathbf{u}|\overline{\mathbf{u}}, \overline{\mathbf{y}}) \tag{2.7}$$

For evaluation at a new query point $\mathbf{u}_*$ the predicted mean of Equation 2.7 is used.

2.3.3 Kernel Functions

In the general expression for a GP, Equation 2.6, $k(\cdot, \cdot)$ represents a covariance function, also referred to as a kernel function. The function models the correlation between each input pair in $\overline{\mathbf{u}}$, $(\mathbf{u}_i, \mathbf{u}_j)$. Once the kernel function is specified, the GP yields a distribution over functions [3]. Valid kernel functions give a positive definite covariance matrix, meaning that it is has the properties of being symmetric and invertible. There are several kernel functions

that are often used to model processes. Kernels can be combined by linear operations. The choice of kernel determines the properties of a GP model and can influence the efficiency of the GP [13].

One of the most used kernel functions is the squared exponential, which is described by Equation 2.8. The squared exponential kernel is also called radial basis function, RBF, or Gaussian kernel. This kernel function results in a smooth prior distribution, which can be observed in Figure 2.4 where the is used. l is the characteristic length scale, which is called the kernel's hyperparameter that are optimized during the fitting of the GP [3]. The length scale determines the smoothness of the function, by setting an "extrapolation limit". Typically, it is not possible to extrapolate more than one length scale away from the data [13].

$$k_{RBF}(\mathbf{u}_i, \mathbf{u}_j) = \exp\left[-\frac{\|\mathbf{u}_i - \mathbf{u}_j\|^2}{2l^2}\right] \tag{2.8}$$

The white noise kernel is possibly the simplest kernel and is stated in Equation 2.9. It allows the GP functions to not have to pass through every data point, by assuming there is some noise in the measurements. The kernel consists of the noise variance σ_n^2 multiplied with the identity matrix of dimension $n_u \times n_u$. The resulting covariance matrix is a diagonal matrix with the variance of each random variable on the diagonal. This implies that all covariances between different variables are zero since the noise is not correlated [13]. The kernel is mainly used with other kernels.

$$k_w(\mathbf{u}_i, \mathbf{u}_j) = \sigma_n^2 \mathbf{I} \tag{2.9}$$

The squared exponential kernel can be combined with the white noise kernel to describe noisy observations, which can be defined the following way [7]

$$K(\mathbf{u}_i, \mathbf{u}_j) = k_{RBF} + k_w = \exp\left[-\frac{\|\mathbf{u}_i - \mathbf{u}_j\|^2}{2l^2}\right] + \sigma_n^2 \mathbf{I}. \tag{2.10}$$

2.3.4 Hyperparameters

As mentioned l and σ_n^2 are referred to as hyperparameters of the kernels. These parameters are learned during the fitting of the GP to the data by maximizing the log marginal likelihood function given in Equation 2.11 [7]. Here Θ is a collective term for the hyperparameters. For the combined kernel in Equation 2.10 Θ would be l and σ_n^2. The numerical values of the hyperparametes can provide insight on how the GP fits the data.

$$LML(\Theta, \overline{\mathbf{u}}, \overline{\mathbf{y}}) = -\frac{1}{2}\overline{\mathbf{y}}^{\mathrm{T}} k(\Theta|\overline{\mathbf{u}})\overline{\mathbf{y}} - \frac{1}{2}\log|k(\Theta|\overline{\mathbf{u}})| - \frac{n}{2}\log 2\pi \tag{2.11}$$

3 Problem Formulation

Building on the theory section, the mathematical equations for the different optimization problems will be presented. First the general steady state RTO problem is introduced followed by the necessary conditions of optimality. Further the modifiers are defined and the modified optimization problem presented for MA. Finally GPs are introduced to represent the plant-model mismatch and included in the MA formulation to propose a version of MA with GP.

3.1 Steady State Optimization Problem

In Equation 2.1 the general RTO optimization problem was presented. The steady state optimization problem for the plant can be mathematically formulated as [7];

$$\begin{aligned} \min_{\mathbf{u}} \quad & \Phi_p(\mathbf{u}) := \phi(\mathbf{u}, \mathbf{y}_p(\mathbf{u})) \\ \text{s.t.} \quad & G_{p,i}(\mathbf{u}) := g_i(\mathbf{u}, \mathbf{y}_p(\mathbf{u})) \leq 0 \;\; i = 1, ..., n_g \\ & \mathbf{u} \in U \end{aligned} \tag{3.1}$$

The decision variables or inputs are $\mathbf{u} \in \mathbb{R}^{n_u}$, while the measured outputs are $\mathbf{y}_p \in \mathbb{R}^{n_y}$. The cost function is $\phi : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}$, which is going to be minimized. The belonging set of constraints for the optimization problem are listed as $g_i : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}$ where i goes from 1 to n_g. The input is limited by the upper and lower bounds $U = \{\mathbf{u} \in \mathbb{R}^{n_u} : \mathbf{u}^{\mathrm{L}} \leq \mathbf{u} \leq \mathbf{u}^{\mathrm{U}}\}$. The subscript $(\cdot)_p$ indicates a quantity related to the plant. The cost and/or constraint functions are typically nonlinear, and then the problem is referred to as a NLP problem [4].

Usually the precise steady state input-output map of the plant, $\mathbf{u} \mapsto \mathbf{y}_p$, is unknown. This could for example be due to unavailable measurements of $\mathbf{y}_p$. Therefore one relies on an approximation given by the best available process model, $\mathbf{F}(\mathbf{x}, \mathbf{u})$, which is often a nonlinear steady state model [5], where $\mathbf{x}$ are the states.

$$\mathbf{F}(\mathbf{x}, \mathbf{u}) = 0 \tag{3.2}$$

$$\mathbf{y} = \mathbf{F}(\mathbf{x}, \mathbf{u}) \tag{3.3}$$

By solving this system of equations the estimated outputs $\mathbf{y}(\mathbf{u})$ can be obtained. Although we would like to minimize Equation 3.1 we are only able to minimize the model:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \Phi(\mathbf{u}) := \phi(\mathbf{u}, \mathbf{y}(\mathbf{u})) \\ \text{s.t.} \quad & G_i(\mathbf{u}) := g_i(\mathbf{u}, \mathbf{y}(\mathbf{u})) \leq 0 \;\; i = 1, ..., n_g \\ & \mathbf{u} \in U \end{aligned} \tag{3.4}$$

Compared with the plant optimization problem in Equation 3.1 the actual measured outputs $\mathbf{y}_p$ are replaced with $\mathbf{y}$ estimated from the process model in Equation 3.3. Note that the cost function ϕ and constraint functions g_i are the same, but since $\mathbf{y}$ changed, the evaluations will be different. One can interpret the model equations as additional equality constraints for the optimization problem 3.4. Due to plant-model mismatch and disturbances, the solutions to Problem 3.1 and 3.4 are usually different and implementation of the optimum of 3.4 as setpoints may result in constraint violation and infeasible operation of the plant.

3.2 Necessary Conditions of Optimality

In order to obtain a feasible point upon convergence, the necessary conditions of optimality, NCO, has to be satisfied. The optimum from the optimization problem in Equation 3.4 can be characterized via the NCO. The set of active constraints, $\mathcal{A}$, at a point $\mathbf{u}$ for the optimization problem in Equation 3.4 is formulated as

$$\mathcal{A}(\mathbf{u}) = \{i \in \{1, ..., n_g\} \mid G_i = 0\}. \tag{3.5}$$

Assume that the Linear Independence Constraint Qualification (LICQ) holds at the optimum $\mathbf{u}^*$, and that the functions Φ and G_i are differentiable at $\mathbf{u}^*$. A vector of all constraints G_i can be denoted $\mathbf{G}$. Then there exists unique Lagrange multipliers $\boldsymbol{\gamma}^* \in \mathbb{R}^{n_g}$ that fulfill the first order KKT conditions[5]. The Lagrange multiplier can be interpreted as a knob that can be modified to adjust the value of $\mathbf{u}$. The Lagrangian function is the sum of the original objective function and a term including the constraint functions multiplied by a vector of the Lagrange multipliers:

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\gamma}) := \Phi(\mathbf{u}) + \boldsymbol{\gamma}^\intercal \mathbf{G}(\mathbf{u}). \tag{3.6}$$

The first order KKT conditions can be formulated as the following;

$$\mathbf{G} \leq \mathbf{0} \tag{3.7}$$

$$\boldsymbol{\gamma}^\intercal \mathbf{G} = 0 \tag{3.8}$$

$$\boldsymbol{\gamma} \leq \mathbf{0}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \frac{\partial \Phi}{\partial \mathbf{u}} + \boldsymbol{\gamma}^\intercal \frac{\partial \mathbf{G}}{\partial \mathbf{u}} = \mathbf{0} \tag{3.9}$$

These three equations make up NCO and are known as primal feasibility, Equation 3.7, complimentarity conditions, Equation 3.8, and dual feasibility conditions, Equation 3.9. If all of the NCO equations holds for $\mathbf{u}$, $\mathbf{u}$ is a feasible point.

3.3 Modifier Adaptation

MA uses first-order corrections to the cost and constraint functions in order to fulfill the NCO of the plant upon convergence. Correction terms depending on the current input are added to the cost and constraint functions of Problem 3.4. At the kth iteration with the operating point $\mathbf{u}_k$ the modified cost function , $\Phi_{m,k}$ can be formulated as

$$\Phi_{m,k} := \Phi(\mathbf{u}) + (\boldsymbol{\lambda}_k^{\Phi})^\intercal (\mathbf{u} - \mathbf{u}_k) \tag{3.10}$$

where $\boldsymbol{\lambda}_k^{\Phi} \in \mathbb{R}^{n_u}$ is the first order modifier for the cost function defined as

$$(\boldsymbol{\lambda}_k^{\Phi})^\intercal := \frac{\partial \Phi_p}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial \Phi}{\partial \mathbf{u}}(\mathbf{u}_k). \tag{3.11}$$

Further, the modified constraint functions $G_{m,i,k}$ can be set up as

$$G_{m,i,k} := G_i(\mathbf{u}) \; + \; \epsilon_{i,k} \; + \; (\boldsymbol{\lambda}_k^{Gi})^\intercal (\mathbf{u} - \mathbf{u}_k) \leq 0, \quad i = 1, ..., n_g \tag{3.12}$$

with the zero order constraint modifier $\epsilon_{i,k} \in \mathbb{R}$ and the first order modifier $\boldsymbol{\lambda}_k^{G_i} \in \mathbb{R}^{n_u}$ given by

$$\epsilon_{i,k} := G_{p,i}(\mathbf{u}_k) - G_i(\mathbf{u}_k) \tag{3.13}$$

$$(\boldsymbol{\lambda}_k^{G_i})^\intercal := \frac{\partial G_{p,i}}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial G_i}{\partial \mathbf{u}}(\mathbf{u}_k). \tag{3.14}$$

Note that the modified cost function does not include a zero order modifier, due to it being a constant term that would not affect the optimal point.

Figure 3.1 shows a graphical representation of the modified constraint function together with the modifiers. It is evident from the figure that the zeroth order modifier $\epsilon_{i,k}$ describes the difference in constraints calculated from the plant measurements and the predicted values from the process model at $\mathbf{u}_k$. Further, the first order modifiers $\boldsymbol{\lambda}_k^{\Phi}$ and $\boldsymbol{\lambda}_k^{G_i}$ correspond to the deviation in the plant gradients and the model gradients. Calculation of the first order modifiers require that the cost and constraint gradients are available at the current $\mathbf{u}_k$. The estimation of the plant gradients at each RTO iteration can be a challenge with MA, especially if the measurements are noisy.

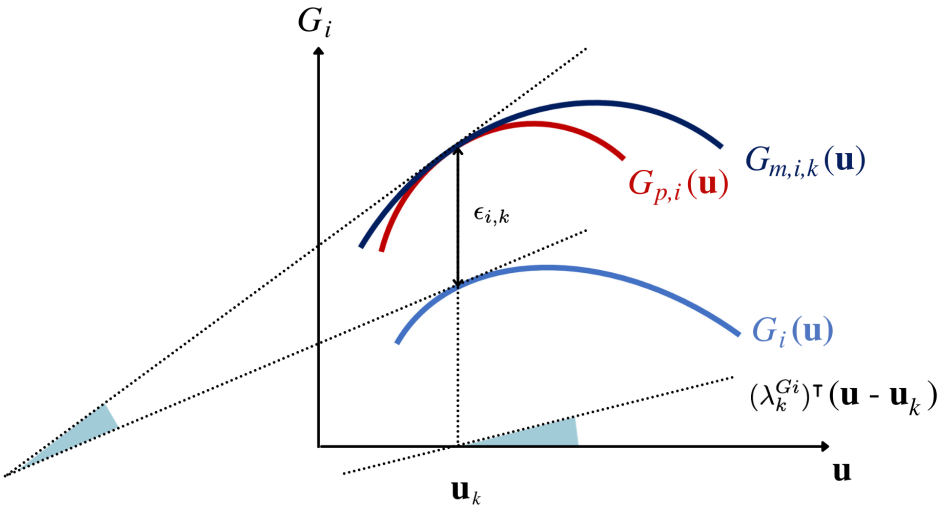

Figure 3.1: Graphical representation of the modified constraint function $G_{m,i,k}$ with the belonging modifiers $\epsilon_{i,k}$, λ_k^{Φ} and $\boldsymbol{\lambda}_k^{G_i}$. Figure based on Figure 1 in [4] and Figure 1 in [5].

At the current point $\mathbf{u}_k$, the succeeding optimal inputs $\mathbf{u}_{k+1}$ are computed by solving the following modified optimization problem:

$$\begin{aligned} \mathbf{u}_{k+1}^* &= \underset{\mathbf{u}}{\operatorname{argmin}}\, \Phi(\mathbf{u}) + (\boldsymbol{\lambda}_k^{\Phi})^\intercal (\mathbf{u} - \mathbf{u}_k) \\ \text{s.t.} \quad & G_{m,i,k} := G_i(\mathbf{u}) + \epsilon_{i,k} + (\boldsymbol{\lambda}_{k+1}^{G_i})^\intercal (\mathbf{u} - \mathbf{u}_k) \leq 0, \\ & i = 1, ..., n_g \\ & \mathbf{u} \in U \end{aligned} \tag{3.15}$$

A filter can be introduced on the new optimal input to prevent too large changes in the input update. This is especially relevant in the presence of noise. The filtered new input is given in Equation 3.16, where $\mathbf{K} = diag(k_1, ..., k_{n_u}) \in \mathbb{R}^{n_u}$ is a diagonal matrix with the filter values $k_i \in (0, 1]$, $i = 1, ..., n_u$, that can be adjusted depending on the magnitude of noise. The filter acts similarly to a trust region, restricting each new input point to move too far from the previous point.

$$\mathbf{u}_{k+1}^* = \mathbf{u}_k^* + \mathbf{K}(\mathbf{u}_{k+1}^* - \mathbf{u}_k^*) \tag{3.16}$$

The same strategy can be used to filter the modifiers in order to reduce the sensitivity to measurement noise. By introducing the filter parameters a_i, b_i and c for the zeroth order, constraint first order and cost first order modifiers respectively, the filtered modifiers for the next iteration can be expressed as the following:

$$\epsilon_{i,k+1} = (1 - a_i)\epsilon_{i,k} + a_i[G_{p,i}(\mathbf{u}_k) - G_i(\mathbf{u}_k)] \tag{3.17}$$

$$(\boldsymbol{\lambda}_{k+1}^{G_i})^\intercal = (1-b_i)(\boldsymbol{\lambda}_k^{G_i})^\intercal + b_i\left[\frac{\partial G_{p,i}}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial G_i}{\partial \mathbf{u}}(\mathbf{u}_k)\right] \tag{3.18}$$

$$(\boldsymbol{\lambda}_{k+1}^{\Phi})^\intercal = (c-1)(\boldsymbol{\lambda}_k^{\Phi})^\intercal + c\left[\frac{\partial \Phi_p}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial \Phi}{\partial \mathbf{u}}(\mathbf{u}_k)\right]. \tag{3.19}$$

The filters limit the distance between sequential iterations. For larger filter parameters the iteration steps become smaller, since a higher ratio of the previous value is included in the updated value. With smaller iteration steps, several iterations will be required to reach the optimum and convergence rate is slowed down.

Gradient estimation for the first order modifiers is often the bottleneck in standard MA. The plant gradients are estimated through finite difference approximations(Appendix A), which are sensitive to noise in the measurement. With increasing number of inputs, gradient estimation becomes challenging. For n_u inputs, one needs to perform n_u step changes of the plant in order to approximate all the input gradients with a first order finite difference. As the plant has to reach steady state for each of the perturbations, gradient approximation is costly since it can be time consuming and result in the plant operating at suboptimal points for longer time periods. To overcome the challenge with approximating the gradients, a scheme that use GP in the modifier calculation will be introduced.

3.4 Modifier Adaptation with Gaussian Processes

GPs can be used to estimate unknown functions. The idea is to use GPs to calculate the differences between the plant and the model for both the cost and constraint function. Once fitted, the GPs can be used to calculate the modifiers. The input data is $\overline{\mathbf{u}}$ and the belonging output data is $\overline{\mathbf{y}}$, as previously introduced. The combined RBF and white noise kernel in Equation 2.10 can be implemented in the GPs, to allow the GPs to explain some variation as noise. The GPs will be adapted recursively, which means that they will be calculated repeatedly for each iteration. The distribution to describe a new prediction point y_* can be written as in Equation 2.7. In this case f are functions that describe the plant model mismatch in the cost and constraint functions given as:

$$f \in \{\Phi_p - \Phi, G_{p,1} - G_1, ..., G_{p,n_g} - G_{n_g}\} \tag{3.20}$$

Consequently the following NLP can be solved in the optimizer of the RTO:

$$\begin{aligned} \mathbf{u}_{k+1}^* \quad &= \quad \underset{\mathbf{u}}{\operatorname{argmin}}\, \Phi(\mathbf{u}) + (\boldsymbol{\lambda}_k^{\Phi_{GP}})^\intercal(\mathbf{u}-\mathbf{u}_k) \\ \text{s.t.} \quad & \quad G_i(\mathbf{u}) + \epsilon_{i,k,GP} + (\boldsymbol{\lambda}_k^{G_{i,GP}})^\intercal(\mathbf{u}-\mathbf{u}_k) \le 0, \\ & \quad i = 1, ..., n_g \\ & \quad \mathbf{u} \in U \end{aligned} \tag{3.21}$$

The GP modifiers $\boldsymbol{\lambda}_k^{\Phi_{GP}}$, $\epsilon_{i,k,GP}$ and $\boldsymbol{\lambda}_k^{G_{i,GP}}$ are found from the GP in Equation 2.7 for the functions in 3.20. Thus the expressions can be stated as the following:

$$(\boldsymbol{\lambda}_k^{\Phi_{GP}})^\intercal = \frac{\partial (GP)_k^{(\Phi_p - \Phi)}(\mathbf{u}|\overline{\mathbf{u}}, \overline{\mathbf{y}})}{\partial \mathbf{u}} \tag{3.22}$$

$$\epsilon_{i,k,GP} = (GP)_k^{(G_{p,i}-G_i)}(\mathbf{u}|\overline{\mathbf{u}}, \overline{\mathbf{y}}) \tag{3.23}$$

$$(\boldsymbol{\lambda}_k^{G_{i,GP}})^\intercal = \frac{\partial (GP)_k^{(G_{p,i}-G_i)}(\mathbf{u}|\overline{\mathbf{u}}, \overline{\mathbf{y}})}{\partial \mathbf{u}} \tag{3.24}$$

In this work the derivatives are approximated by finite differences in Equation 3.22 and 3.24 (Appendix A). The filter on the optimal inputs in Equation 3.16 can be used for the GP scheme, since it works as a trust region for the input update. This is useful to keep the evaluations near the region where the GP has its data. However the filter on the modifiers should not be necessary, as the GP handle noise in the measurements internally.

4 Case Study: Williams-Otto Reactor

In this section a case study with Williams-Otto reactor will be presented. The Williams-Otto reactor was proposed in 1960 to use for direct comparison of computer control, in order to expose limitations and performance[14]. The reactor is a hypothetical reactor with fictitious chemical compounds.

We will use a plant and process model with different model structures. The plant model will be used to provide measurements for the process model. Consequently, there is structural plan-model mismatch for the system. The reactor is an ideal continuous stirred tank reactor, that produces the desires products P and E in addition to an undesired byproduct G.

Figure 4.1 shows a simple illustration of the reactor. The reactor feed consists of F_A and F_B, which are continuous flows. The feed flows are pure and consists of 100% A and B respectivly. F_A is constant and has a value of 1.8275 kg/s. F_B is varying and is one of the controlled variables or inputs for the system. The second input is the reactor temperature, T_R. The reactor temperature influences the reaction rates. Thus the input vector becomes $\mathbf{u} = [F_B, T_R]^\intercal$.

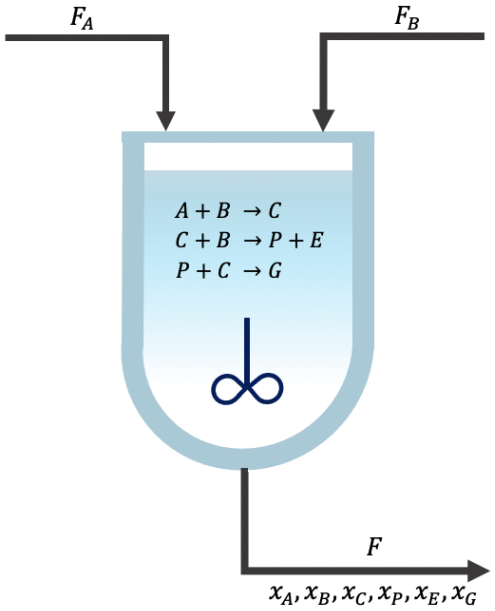

Figure 4.1: Illustration of the Williams-Otto reactor with feed streams F_A and F_B, outlet stream F and the plant reactions.

The actual reactions that take place in the reactor are presented in Equation 4.1-4.3. The model describing the complete reactor system will be referred to as the plant. Reaction 4.1 and 4.2 produce the desired products and reaction 4.3 produce the byproduct. Each of the reactions has a specific reaction rate constant k_1, k_2 and k_3.

$$A + B \xrightarrow{k_1} C \tag{4.1}$$

$$C + B \xrightarrow{k_2} P + E \tag{4.2}$$

$$P + C \xrightarrow{k_3} G \tag{4.3}$$

We use a simplified model for the process model, as if we had incomplete understanding of the system. Therefore it has a simplified reaction scheme as described in Equation 4.4-4.5. The

simplification implies that there is no production of the intermediate product C, which results in changed mass balances and reaction rate constants, k_1^* and k_2^*. Due to the simplification, the calculated optimum operating points for the plant will be different from the optimum operating points calculated from the process model.

$$A + 2B \xrightarrow{k_{1^*}} P + E \tag{4.4}$$

$$A + B + P \xrightarrow{k_{2^*}} G \tag{4.5}$$

4.1 Plant and Model Equations

The plant and process model have been used in various papers and the model equations used here are found in[15], except we use mass basis and not molar basis. The reaction rate constants can be calculated through Arrhenius equation, given in Equation 4.6[16]. A_i is the pre-exponential factor and $E_{a,i}$ is the activation energy in K associated with the reaction i. T_R is the reactor temperature in °C.

$$k_i = A_i e^{-E_{a,i}/(T_R+273.15)}, \quad i = 1, 2, 3, 1^*, 2^* \tag{4.6}$$

The system equation for the plant and the process model are used to calculate the states for a given input. The states for the plant, $\mathbf{x}_p$, are the mass fractions of the different components in the reactor, $[x_A, x_B, x_C, x_E, x_P, x_G]^\intercal$, while for the process model it is the same mass fractions except x_C, $\mathbf{x} = [x_A, x_B, x_E, x_P, x_G]^\intercal$. The reaction rates for the plant reactions can then from rate expressions be written as[16];

$$r_1 = k_1 x_A x_B M_t \tag{4.7}$$

$$r_2 = k_2 x_B x_C M_t \tag{4.8}$$

$$r_3 = k_3 x_C x_P M_t. \tag{4.9}$$

M_t denotes the total mass in the reactor, which is constant and equal to 2105.2 kg[17]. Similarly the reaction rates for the model are given as

$$r_{1^*} = k_{1^*} x_A {x_B}^2 M_t \tag{4.10}$$

$$r_{2^*} = k_{2^*} x_A x_B x_P M_t. \tag{4.11}$$

The total flow rate F is the sum of the reactant flow rates, $F_A + F_B$. Due to the conservation of mass and closed system, the flow rate out of the reactor equals F. Finally, the steady state mass balances for the plant can be defined from mass conservation laws and expressed as follows,

$$\begin{aligned} Fx_A &= F_A - r_1 \\ Fx_b &= F_B - r_1 - r_2 + Fx_b \\ Fx_C &= 2r_1 - 2r_2 - r_3 \\ Fx_E &= 2r_2 \\ Fx_P &= r_2 - 0.5r_3 \\ Fx_G &= 1.5r_3. \end{aligned} \tag{4.12}$$

For the process model the steady state mass balances are given as,

$$\begin{aligned} Fx_A &= F_A - r_{1^*} - r_{2^*} \\ Fx_B &= F_B - 2r_{1^*} - r_{2^*} \\ Fx_E &= 2r_{1^*} \\ Fx_P &= r_{1^*} - r_{2^*} \\ Fx_G &= 3r_{2^*}. \end{aligned} \tag{4.13}$$

The numerical values of the kinetic constants are given in Table 4.1.

Table 4.1: Kinetic parameters for the plant and model reactions. Values from the plant are obtained from [6] and model parameters were decided through personal communication.

Reaction	**Pre-exponential factor**			**Activation energy**		
	Parmeter	Value	Unit	Parameter	Value	Unit
1	A_1	$1.6599{\cdot}10^6$	1/s	$E_{a,1}$	6666.7	K
2	A_2	$7.2117{\cdot}10^8$	1/s	$E_{a,2}$	8333.3	K
3	A_3	$2.6745{\cdot}10^{12}$	1/s	$E_{a,3}$	11111	K
1*	A_{1^*}	0.04979	1/s	$E_{a,1^*}$	6513.6	K
2*	A_{2^*}	0.01832	1/s	$E_{a,2^*}$	11111.3	K

4.2 Optimization Problem

The objective for the reactor is to maximize the economical plant profit by manipulating the inputs $[F_B, T_R]^\intercal$. The cost function is presented in Equation 4.14 with the belonging constraints and consists of a sum of product profit subtracted by the cost of the feed flows. Note that the unit of the cost function is \$/s. The cost of utilities are neglected and P_i is the price for the component i. There are two constraints that limits the weight fractions on two states, x_A and x_G. In addition there are upper and lower bounds on the inputs.

$$\begin{aligned} \max_{F_B, T_R} \quad & \Phi_p = P_P x_P F + P_E x_E F - P_A F_A - P_B F_B \\ \text{s.t.} \quad & g_{p,1} = x_A - 0.12 \leq 0, \\ & g_{p,2} = x_G - 0.08 \leq 0, \\ & F_B \in [4, 7], \\ & T_R \in [70, 100] \\ & \text{Eq.4.12} \end{aligned} \tag{4.14}$$

In addition to the inequality constraints and the bounds, the plant equations, Equation 4.12, have to be fulfilled to optimize the plant profit. Note that although we want to optimize the plant we only have the process model available, 4.13 (see Section 3.1). The prices for the different chemical compounds are presented in Table 4.2. The measurements are obtained using the plant equations 4.12.

Table 4.2: Prices for the products P and E, and the feed reactants A and B [7].

Price	P_P	P_E	P_A	P_P	Unit
Value	1043.38	20.92	79.23	118.34	[\$/kg]

5 Results and Discussion

Simulation of the standard MA and the MA with GP schemes for optimizing the Williams-Otto reactor is carried out for different cases, studying the effect of measurement noise. Since we measure the weight fractions, $\mathbf{x} = [x_A, x_B, x_C, x_E, x_P, x_G]^\intercal$, which are the states, we get that $\mathbf{x} = \mathbf{y}$. We assume that these measurements are exposed to the same random noise, which is assumed to be independent and identically distributed by a normal distribution with zero mean and some standard deviation, σ.

The noise in the constraints will be equal to the measurement noise since the constraints only contain one measurement (Equation 4.14). The actual noise of the cost function requires some calculation, since it includes two measurements. From statistical calculation rules [18], the following expression for the noise in the cost function is

$$\sigma_{\Phi} = \sqrt{(P_P F_A \sigma)^2 + (P_E F_A \sigma)^2}. \tag{5.1}$$

The first term represents the measurement noise in x_P and the second term is the noise in x_E. By including the numerical values of P_P and P_E in Table 4.2 and the constant value of $F_A = 1.8275$ kg/s, it is evident that the magnitude of the noise in the cost function is significantly larger than for the constraints. Consequently, one can expect that increase in the measurement noise would have large impact on the cost calculation.

5.1 The Case Studies

The influence of noise on standard MA is studied by changing the noise in the system. The first case is with zero noise in the plant measurements. Further, constant noise for cost function and constraint function is added to the system and solved. The magnitude of noise is varied in order to find the limit where the algorithm does not converge to the optimum. Further, the actual cost noise from Equation 5.1, is introduced. Finally, alternative solutions for tackling convergence challenges in the presence of noise is suggested for the standard MA. The algorithm for the standard MA implementation for Williams-Otto reactor is outlined in Algorithm 1.

In the MA and GP scheme, a similar approach will be used, and we will also investigate different formulations of the GP. In the first case there is no measurement noise. Secondly, four different noise levels with noise of the cost function from Equation 5.1, are investigated for three versions of the GP scheme. In the first version, the current operating point $\mathbf{u}_k$ and system response is added to the GP training data at every iteration. The other versions provide more training data. In the second version an additional point for $0.95 \cdot \mathbf{u}_k$ is included in the training data at every iteration. In the third version an extra response at the current operating point is provided, i.e. we add two identical and independently distributed samples to the training date in each iteration. The different versions are studied in order to detect the best performance. An outline of the algorithm for the MA with GP is presented in Appendix 2.

In all simulations $\mathbf{u}_0$ is $[7, 70]^\intercal$, and the maximum number of iterations is 20. Plant gradients are calculated with finite differences (Appendix A), where the perturbation in $\mathbf{u}$, h, is set to 10^{-4} for the standard MA. In the cases with GP, the gradients of the first order modifiers in Equation 3.22 and 3.24 are calculated with $h = 10^{-8}$. The filter parameters from Equation 3.16, 3.17, 3.18 and 3.19 are set to $k_i = 0.4, i \in 1, 2$ for the input filter and $a_i = b_i = c = 0.6, i \in 1, 2$ for the modifier filters. The filters on the modifiers are only implemented for the standard MA. Further, the initial training points for the GPs include $\mathbf{u}_0$ and 4 arbitrarily chosen surrounding points of $\mathbf{u}_0$, together with the respective output evaluations for the three GPs (Equation 2.7 and 3.20). The global optimum for the plant is $\mathbf{u} = [4.3894, 80.4948]^\intercal$, which is the desired ending point the algorithms should converge to. The model optimum is $\mathbf{u} = [4.5684, 100]^\intercal$, which substantiate the structural plant-model mismatch. These optimums are obtained by optimization of the plant and the model separately in Problem 4.14.

In the following sections the programming environment is described and the results of the different schemes are presented and discussed. Note that for all the plots with presence of

noise, one random run of the relevant scheme is presented as an example. Due to the noise being random, there are in practise an infinite number of scenarios, and the presented plots can be seen as a snapshot of the schemes behavior for the current settings.

5.2 Programming Environment

For implementation and simulation of the different optimization schemes, Python version 3.9.9 was used. CasADi[19] was used as a tool for setting up and solving the optimization problems, with IPopt[20] used for the optimization. Further, the numpy package[21] was used for mathematical programming, and Sklearn[22] was used for the GPs.

5.3 Standard Modifier Adaptation without Noise

The first simulation is the standard MA with zero noise. The calculated optimal values of the inputs F_B^* and T_R^*, the profit, which is negative cost $-\Phi_p^*$, and the weight fractions with constraints, x_A^* and x_G^* are presented in Table 5.1. It is evident that the scheme converges to the global optimum, where both of the inequality constraints are active. However, there is a small deviation from the true plant optimum in T_R^* of approximately 0.01 °C. This could be due to the error in the finite difference approximation of the gradients (Appendix A). For this first simulation, several plots will be provided to study how the cost, inputs and constrained variables develop towards the optimum.

Table 5.1: The optimal values of the flow rate of B, F_B^*, the reactor temperature, T_R^*, the profit or negative cost, $-\Phi_p^*$, and the constrained weight fractions, x_A^* and x_G^*, from simulation of the standard MA without noise.

Parameter	F_B^* [kg/s]	T_R^* [°C]	$-\Phi_p^*$ [$/s]	X_A^* [−]	X_G^* [−]
Value	4.3894	80.5057	75.9075	0.1200	0.0800

Figure 5.1 shows a plot of the calculated inputs for every iteration, where reactor temperature, T_R, is plotted as a function of flow rate of B, F_B. Each blue star represents the one $\mathbf{u}_k$ and the dotted line shows the order of the iteration points. The final input is marked with a red star and is denoted $\mathbf{u}_k^f$. The initial input, $\mathbf{u}_0$, is the point in the lower right corner. It can observed that the system moves closer to the optimum with every iteration, and by approximately 9 iterations the plant the optimum is reached. The red lines represents the constraints, where the lower is g_1 and the upper is g_2. The feasible region is the light blue coloured area in the plot. From the plot it is visible that the intersection of the constraints is the plant optimum, and the only point within the operating region where both constraints are active.

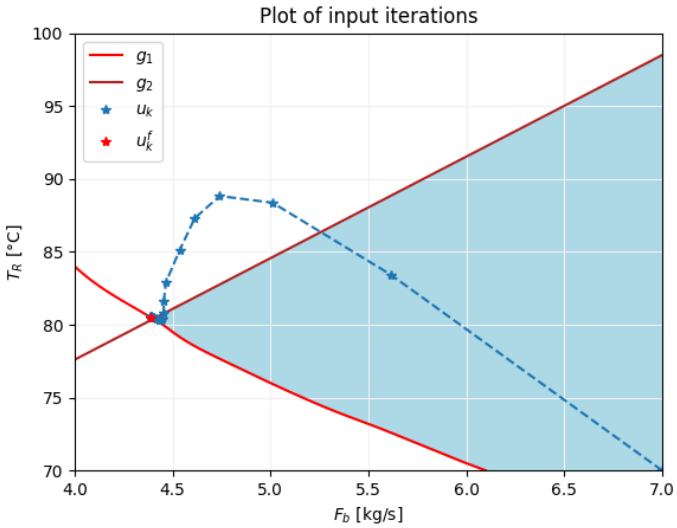

Figure 5.1: Plot of input iterations for standard MA without noise. Each star represents one iteration and the dotted line shows the order of the iterations. The final iteration is labelled u_k^f. g_1 and g_2 are the constraints that bound the feasible region indicated with the blue area.

The development of the plant profit, $-\Phi$, together with the inputs are plotted against iteration number in Figure 5.2. It can be observed that the plant profit is at its highest at iteration 5, but due to constraint violation, this is not the optimum. The profit converges to the optimal value of 75.9075\$/s. Due to a negative initial profit, the inputs change drastically the first iterations. F_B change from 7 to 5 kg/s in two iterations, before it eventually reach the optimum around 4.4 kg/s. In T_R an overshoot can be observed for the first iterations. Note that the information in the input plots can be observed in Figure 5.1 as well, where the stars illustrate the iterations.

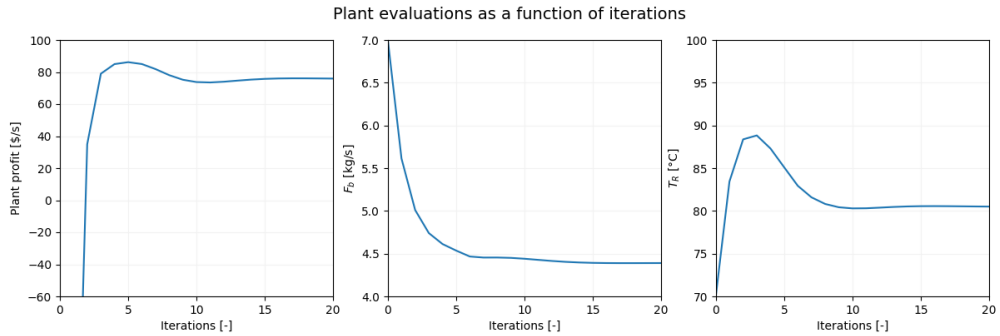

Figure 5.2: Plant profit, T_R and F_B plotted against iterations for standard MA without noise.

Figure 5.3 displays the constrained weight fractions, x_A and x_G, as a function of iterations Constraint g_2 is violated from iteration 2 to 7. One explanation is that the plant-model mismatch can lead to calculations that violates the constraint. From Figure 5.1 it can be observed that model optimum at $\mathbf{u} = [4.5684, 100]^\intercal$ would violate constraint g_2. This explains the constraint violation before convergence. Both of the weight fractions reach the constraint limits at the final iterations, as already observed in Figure 5.1 and Table 5.1. It can be noted that Figure 5.1 contains most of the information that is obtained in Figure 5.2 and 5.3, except the plant profit development. Since we are going to study the ability of convergence of different schemes in the presence of noise, convergence can be studied in the input iteration plot and the profit development is not necessary. Therefore plots corresponding to Figure 5.1 are presented

in in the following sections.

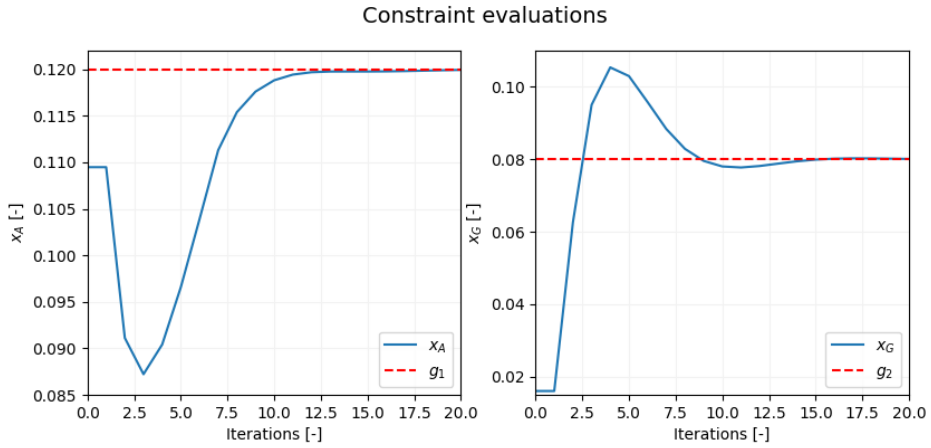

Figure 5.3: Weight fraction x_A and x_G plotted against iterations for standard MA without noise. The constraint values are indicated with red dotted lines.

5.4 Standard Modifier Adaptation with Noise

Noise is introduced on all the measured parameters in the system. Since the gradient calculations in the standard MA scheme is highly sensitive to noise, the initial noise has a small value of $8 \cdot 10^{-8}$. This equals a $\pm 10^{-6}$ deviation on the constraints which is an exceptionally small error and represent 0.0001% of the constraint value in g_2 (Equation 4.14). At first noise in the constraints is assumed to be equal to the noise in the cost function. As discussed previously this is unrealistic but serves as a base case. Further the noise is increased by a factor of 10 to $8 \cdot 10^{-7}$. These two cases are termed constant noise. Finally the same noise levels are introduced with actual cost noise calculated from Equation 5.1. The resulting simulations are presented with the iteration trajectories of $T_R(F_B)$ in Figure 5.4.

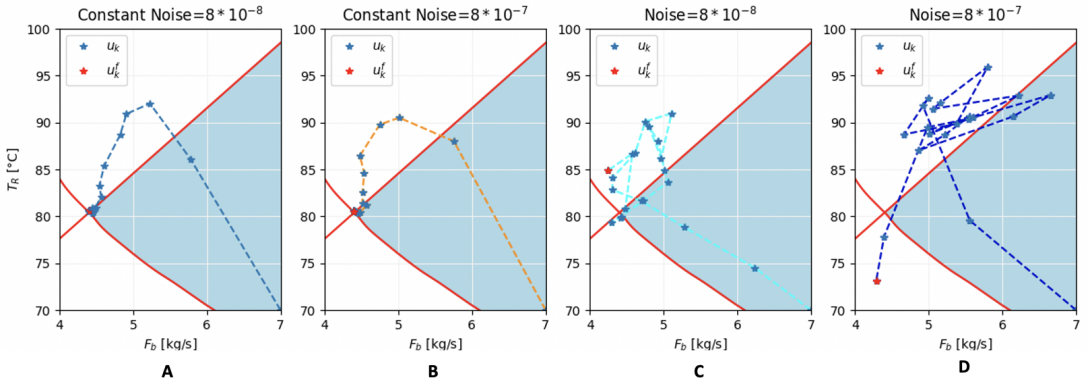

Figure 5.4: Plot of standard MA with 4 different noise levels. Plot A and B have equal noise for constraint and cost function. Plot C and D have cost noise calculated from Equation 5.1.

The first simulation with equal noise for constraint and cost functions with a value of $8 \cdot 10^{-8}$, converges to the optimum as observed in plot A in Figure 5.4. The trajectory is relatively similar to the one without noise in Figure 5.1, except from the path being less smooth. At iteration 9, the input value is approximately at the optimum. When the noise is increased to $8 \cdot 10^{-7}$ in the plot B, the trajectory is similar and reach the region of the optimum in the same number of iterations. Introducing cost function noise calculated from Equation 5.1, results in the standard MA being unable to converge to the optimum. The reason is that when the actual

cost noise is calculated, it introduces a significantly larger error to the system. The increase in the cost noise is a challenge for the MA and the consequence is that the optimization fails. This holds for both noise levels equal to $8 \cdot 10^{-8}$ and $8 \cdot 10^{-7}$, and can be observed in plot C and D in Figure 5.4.

A measure that can be implemented to improve the performance of the standard MA in presence of noise, is to increase the filter parameters. Initially the filter parameters are $k_i = 0.4, i \in \{1, 2\}$ for the input filter and $a_i = b_i = c = 0.6, i \in \{1, 2\}$ for the modifiers. By increasing the filter parameters on the inputs, it ensures that the sequential input iterations are close to each other and prevents large iteration steps. The filter on the modifiers acts as a running average, hence the influence of outliers or high noise measurements are reduced. In Figure 5.5 four plots with increasing filter values with measurement noise equal to $2 \cdot 10^{-7}$ and actual cost noise, are displayed.

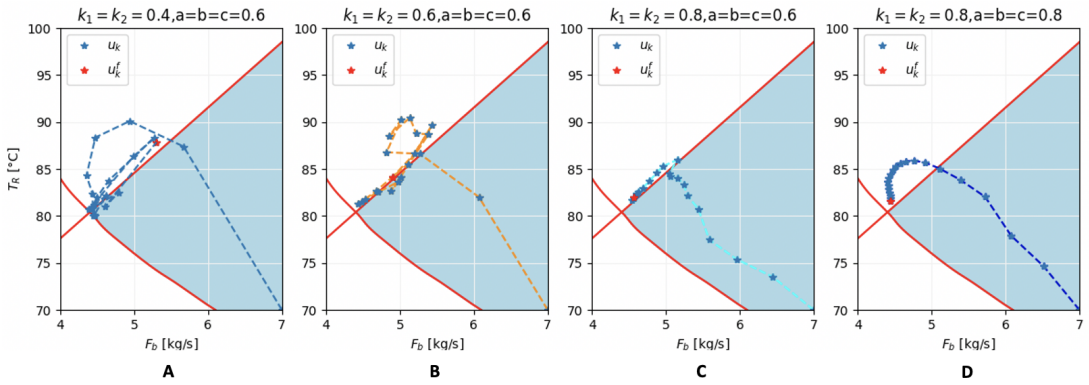

Figure 5.5: Plot of input iterations for standard MA with different filter parameters. k_1 and k_2 are filter parameters for the inputs, a, b and c are filter parameters for the modifiers. The measurement noise level is equal to $2 \cdot 10^{-7}$ for all plots.

Plot A in Figure 5.5 has the initial filter settings, that were used in the simulations in Figure 5.4. However, the noise level is different. It can be observed that the MA is unable to converge for the initial filter settings. In plot B, the filter parameters on the inputs are increased to 0.6 and the noise is unchanged, but the optimum is still not reached. When the input filter parameters are increased to 0.8, the scheme seems to move closer to the optimum in plot C. However, with a filter this large on the inputs, the iterations are restricted to move very little for each step. Therefore the MA would need several iterations in order to converge to the optimum. Still, for the plot C with $k_i = 0.8$ and $a_i = b_i = c = 0.6$ it can be observed that the last iterations jump back and forth the final point and does not seem to move closer to the plant optimum. Thus, this scheme would likely not converge to the plant optimum even with several iterations.

Finally in plot D, the filter parameters on the modifiers are increased to 0.8. The result is a trajectory that moves slowly but steadily towards the optimum. The final iteration does not reach the optimum, but with an increased number of iterations it would have converged to the correct point. Consequently, the simulations prove that the performance of the standard MA can be improved by heavy filtering. However, the increase in the filter parameters cause a slower convergence rate compared with the case without noise. It should be emphasized that the noise used here of $2 \cdot 10^{-7}$ is still very small, and for a small increase in the measurement noise the standard MA would not converge even with the heavy filtering.

5.5 Modifier Adaptation with Gaussian Processes without Noise

Gaussian processes are introduced to the optimization problem according to Equation 3.21 with the plant-model mismatch described by a GP for the cost and constraint functions. In the first simulation, the system is not exposed to measurement noise. The training data provided for the GPs is a vector consisting of all previous $\mathbf{u}_k$, in addition to the initial $\mathbf{u}_0$ and four surrounding points. The final values of the inputs, the plant profit and the constrained weight fractions are presented in Table 5.2 and the input trajectory is presented in Figure 5.6.

Table 5.2: The optimal values F_B^*, T_R^*, $-\Phi_p^*$, and x_A^* and x_G^*, from simulation of MA with GP without noise.

Parameter	F_B^* [kg/s]	T_R^* [°C]	$-\Phi_p^*$ [$/s]	X_A^* [−]	X_G^* [−]
Value	4.3894	80.4946	75.8105	0.1200	0.0800

Compared with the optimal values for the standard MA in Table 5.1, all values except F_B^* and $-\Phi_p^*$ are equal. The values for F_B^* and $-\Phi_p^*$ are lower than for the standard MA case, but closer to the plant optimum. The reason for the deviation could be the inaccuracy in the finite difference calculation of the gradients. The perturbation for the finite differences are set to 10^{-8} for MA with GP compared to 10^{-4} for standard MA. Gradient calculation is involved in all the first order modifiers, and due to the decrease in the perturbation, the accuracy of the MA and GP is expected to be higher. Consequently the optimum obtained for MA with GP is closer to the plant optimum.

Figure 5.6 validates that the scheme converges to the optimum. From the plot it can be observed that the system is close to optimum after 9 iterations which is similar to the standard MA without noise. Compared with the standard MA, the trajectory is more "turbulent" with larger steps in varying directions for the first iterations. This can be explained by the absence of the filters on the modifiers that restrict the change in the cost and constraint gradients and allow for more drastic directional changes from one iteration to the next. As for the standard MA, constraint g_2 is violated due to the plant-model mismatch, before the optimum is reached at the constraint.

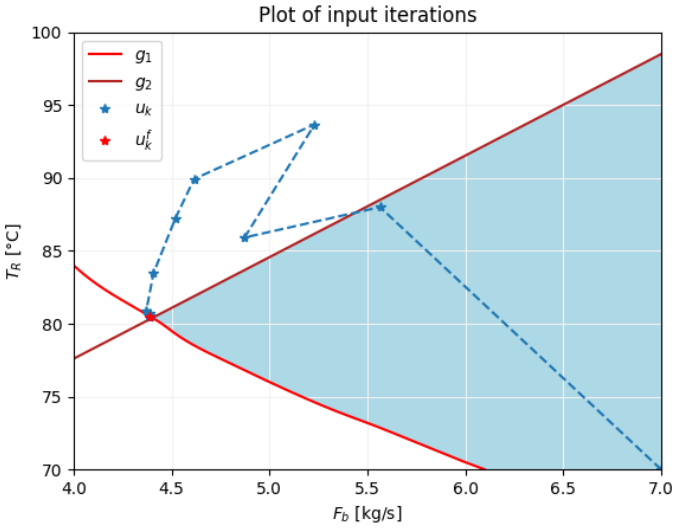

Figure 5.6: Plot of input iterations for MA with GP without noise.

5.6 Modifier Adaptation with Gaussian Processes with Noise

In this section the performance of the MA with GP scheme in presence of noise is studied. It was observed that the standard MA failed to converge for both noise levels $8 \cdot 10^{-8}$ and $8 \cdot 10^{-7}$ when the actual cost noise was implemented. Since the GP is expected to handle noise well, the initial noise level introduced is the highest noise level used for the standard MA, $8 \cdot 10^{-7}$. Further the noise level is increased to $8 \cdot 10^{-4}$, $8 \cdot 10^{-3}$ and finally $1.6 \cdot 10^{-2}$. The noise levels are labeled A, B, C and D respectively in the presented plots and tables. Compared with the constraint value of g_2 the different noises represent a noise percentage on the constraint of $0.001\%, 1\%, 10\%$ and 20%. For all cases in this section, the noise for the cost function is calculated from Equation 5.1.

Three different versions of GP implementation are studied: a base case, providing minimal training data to the GP, a version with one additional steady state input and a version with one additional identically and independently distributed (iid) estimate. In the base case, the training data consists of the initial training points in addition to the calculated $\mathbf{u}_k$s for the previous iterations. Thus, the number of training data points increase with one for each iteration, providing the GP with more information. This was the case for the MA with GP without noise. In order to provide the GP with more information from the beginning, version two and three are introduced, where the number of training data points are the double compared with the base case. In version two the training data is supplied with an extra perturbed point at each iteration using 95% of the current point. This factor was arbitrarily chosen. In the final version, the training data consists of two iid estimates of the same operating point at each iteration. The performance of the MAs with GP are investigated and discussed in the following sections, by studying the $T_R(F_B)$ trajectory and the numerical values of the RBF length scale, l, and the white kernel noise, σ_n^2.

5.6.1 Base case

In Figure 5.7 the input trajectories are presented for the base case. For the smallest noise level in Figure 5.7 A, it is evident that the scheme reach the optimum quickly. After 7 iterations the input is close to the optimum, which is faster than both the standard MA and the MA with GP without noise. However, this may be due to advantageous random noise realisations. Note that the noise level is equal to the largest noise level introduced to the standard MA (Figure 5.4 D), where the optimizer completely failed. Thus, the MA with GP implementation has already outperformed the standard MA, by being able to converge to the optimum at noise level $8 \cdot 10^{-7}$. To test how much noise the MA with GP can handle, the higher noise levels are introduced.

In Figure 5.7 B, the noise is significantly increased by a factor of 10^3 to $8 \cdot 10^{-4}$. The converges to the plant optimum is for this noise level as well, but use several iterations. With noise level increased to $8 \cdot 10^{-3}$ in plot C, the scheme converges to a point that is not the true optimum, as constraint g_1 is violated. When doubling the amount of noise to $1.6 \cdot 10^{-2}$, the optimization fails and the scheme is not converging, as displayed in plot D.

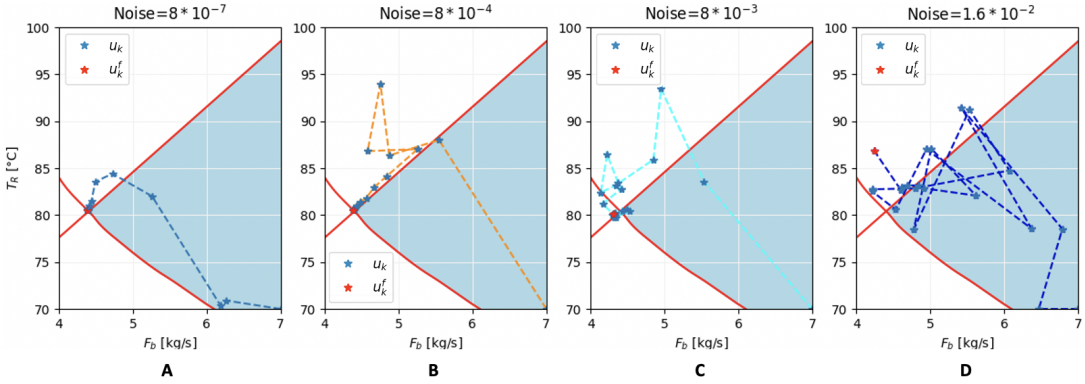

Figure 5.7: Plot of input iterations for the base case of MA with GP for 4 different noise levels.

The final hyperparameters of the kernel function for each noise level and GP are presented in Table 5.3. The values can explain the behaviour in Figure 5.7. As observed for plot A, the MA converges quickly and the noise level is small. The belonging hyperparameters are displayed in the first row of Table 5.3. All the white noise levels are equal and have a value of 10^{-6}. The small value reflects that most of the variations in the data are explained by the RBF kernel and not by the white noise kernel. This is as expected, since the measurement noise is small. The magnitude of the measurement noise and the white noise cannot be compared directly, because the GP is scaling the input and output. However, changes in the white noise versus the measurement noise can be compared. The RBF length scales, l, are 4.46, 3.6 and 4.04 for the cost GP, g_1 GP and g_2 GP respectively. Since this case converge to the optimum and has sensible white noise, it is reasonable to use these length scales as a benchmark to compare the other length scales with.

For the noise level in plot B it can be observed that the magnitude of the white noise increase proportionally with the increased measurement noise. Both increase by a factor of 10^3, which indicate that the GPs explain a realistic ratio of variation in the training data with noise. The length scales are slightly changed, but not drastically. Consequently, the GPs are expected to have the ability to explain the relationship in the input and output data. The result is that the optimizer succeeds to converge to the optimum as observed in plot B in Figure 5.7.

With measurement noise level equal to $8 \cdot 10^{-3}$, the scheme converges to a point that is close to, but not at the optimum (Figure 5.7 C). This can be explained by the hyperparameters for the GP on g_1. Here it can be noted that the white noise has decreased to 10^{-6}, which is significantly smaller than expected. One would expect the white noise to increase from B to C. Thus, the GP tries to explain more of the variations in the data with the noise-free function. The small length scale with a value of 0.0605 contributes to increased complexity in the GP model. This complexity can lead to overfitting of the data, which is likely the case here. The result is that the GP for constraint function g_1 does not provide a sufficient model for the optimizer, and leads to the constraint being violated at the point of convergence. Therefore the final point in plot C is not the plant optimum.

The optimization fails for the noise level in plot D. What was observed at noise level C for constraint g_1, can be seen for constraint g_2 here. With the same reasoning, the GP attempts to explain very little of the variations in the data with noise, and therefore the constraint is violated for at the final point, u_k^f.

Table 5.3: Final hyperparameters, RBF length scale, l, and white noise, σ_n^2, for the three GPs for the base case MA with GP.

		Gaussian process hyperparameters					
Noise		$(GP)^{\Phi_p-\Phi}$		$(GP)^{g_{p,1}-g_1}$		$(GP)^{g_{p,2}-g_2}$	
Level	Value	l	σ_n^2	l	σ_n^2	l	σ_n^2
A	$8 \cdot 10^{-7}$	4.46	10^{-6}	3.6	10^{-6}	4.04	10^{-6}
B	$8 \cdot 10^{-4}$	6.02	$1.02 \cdot 10^{-3}$	2.06	$9.11 \cdot 10^{-3}$	3.71	$1.27 \cdot 10^{-3}$
C	$8 \cdot 10^{-3}$	9.0	0.153	0.0605	10^{-6}	1.72	0.0724
D	$1.6 \cdot 10^{-2}$	9.63	0.431	2.96	0.345	0.077	$5.59 \cdot 10^{-9}$

5.6.2 GP Version 2: Additional Training Data at New Point

For the second MA with GP version an extra perturbed point is added at each iteration using 0.95*current point, to provide the GP with more training data to improve the performance. We are giving the GP more information in the evaluated regions so that the prediction may be better. In practice, this case can be interpreted as calculating two $\mathbf{u}_k$ at each iteration, which requires more time to detect the steady state for both points. The same measurement noises as for the base case are introduced, and the resulting input trajectories are presented in Figure 5.8 and the belonging final hyperparameters in Table 5.4.

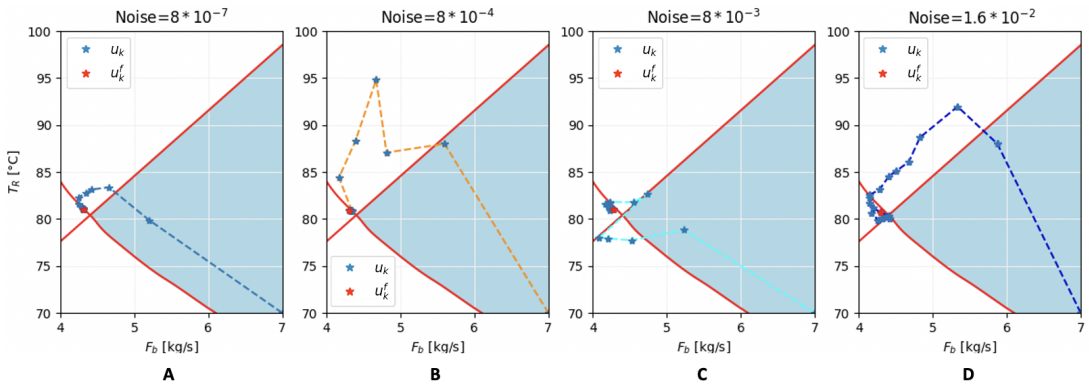

Figure 5.8: Plot of input iterations for the second version of MA with GP for 4 different noise levels.

The first thing that can be noticed in Figure 5.8 is that all the simulations seems to converge to a point that deviates from the optimum, regardless noise level. By looking at the hyperparameters, it is evident that many of the GPs are not representing the variations in the data well. From Table 5.4 it can be observed that the white noises for noise level A are too large for all of the GPs. In the base case all σ_n^2 were 10^{-6} for this noise level. This indicates that the GP is trying to incorrectly describe variations as noise, which results in an inaccurate prediction. Further, the length scale for the constraint g_2 GP is extremely small, which results in a overspecified model for noise level A. It should also be noted that the length scale for constraint g_1 is very large for noise level D. This allows for larger extrapolations steps and a "smoother" function, but can lead to poor accuracy on the described input-output-relation.

Table 5.4: Final hyperparameters, RBF length scale, l, and white noise, σ_n^2, for the three GPs for the second version of MA with GP.

		Gaussian process hyperparameters					
Noise		$(GP)^{\Phi_p-\Phi}$		$(GP)^{g_{p,1}-g_1}$		$(GP)^{g_{p,2}-g_2}$	
Level	Value	l	σ_n^2	l	σ_n^2	l	σ_n^2
A	$8 \cdot 10^{-7}$	6.59	0.598	3.18	$3.16 \cdot 10^{-3}$	$3.71 \cdot 10^{-4}$	0.107
B	$8 \cdot 10^{-4}$	10.3	0.214	1.92	$7.86 \cdot 10^{-3}$	2.87	0.0451
C	$8 \cdot 10^{-3}$	2.63	0.547	5.52	0.465	6.92	0.263
D	$1.6 \cdot 10^{-2}$	8.79	0.719	1190	1.02	5.52	0.574

In conclusion, this version with an additional steady state point does not improve the performance of the MA with GP. One reason could be that the additional point is chosen in an arbitrary direction.

When choosing between the two versions of MA with GP studied so far, the base case would be the better implementation because it was able to converge to the plant optimum whereas the second version did not. Choosing the point direction of the additional point with care could be one method to improve the performance of this version. One can for example choose a decreasing direction, i.e. the gradient's direction. This could be advantageous if the plant-model mismatch is large. Another option is to relocate the extra point closer to the existing location. The GP can get direct information on the noise level by providing two estimates of the same point. This will be studied in the next section.

5.6.3 GP Version 3: Two iid Estimates of Plant Response at $\mathbf{u}_k$

In this version of MA with GP, the GPs are given two iid estimates of the same operating point at each iteration. In implementation this implies to calculate the plant-model cost and constraint deviation subject to random noise twice for the same input $\mathbf{u}_k$. If implemented in a real plant it is important to ensure that the two measurements are independent. This yields two different estimates of the output training data with equal values for the belonging input training data at each iteration. Thus the number of datapoints are doubled compared with the base case, and equal compared with the second version. Once again, the same measurement noises as for the base case are introduced, and the results are presented in Figure 5.9 and Table 5.5.

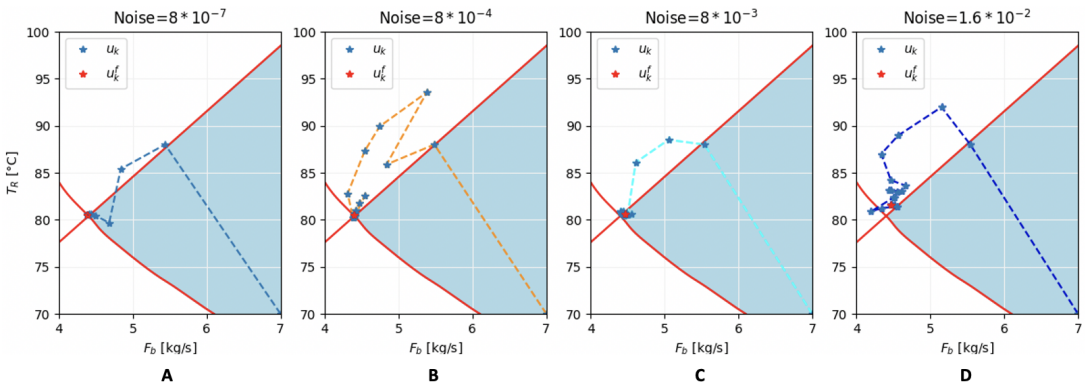

Figure 5.9: Plot of input iterations for the third version of MA with GP for 4 different noise levels.

For the first and smallest measurement noise level, displayed in plot A in Figure 5.9, the scheme converges efficiently to the plant optimum. After only 6 iterations, the optimum is reached, which is the fastest convergence out of all the cases studied. Again, this could be due to fortunate random noise. The satisfactory performance is validated by the values of the hyperparameters. The white noises are equal to the values of the base case with noise level A (Table 5.3). The length scales deviate slightly from the ones obtained as benchmarks in the base case, but are of the same magnitude and much closer than in the second version.

In plot B the scheme converges to the plant optimum. The changes in white noise is as expected for the cost GP and constraint g_2, increasing the same order of magnitude as the measurement noise. For constraint g_1, the white noise is roughly one decimal larger, and the length scale smaller. However, it does not seem to affect the convergence ability, and therefore it is not significant. Further, the plant converges to the optimum for the noise level in plot C. This is reflected in the hyperparameters, that does not have any remarkable changes. The number of iterations for the plant to reach the region of the optimum is low, approximately only 4 iterations. However, observe that some iterations seems to oscillate around the optimum. Regardless, it is evident that this version outperforms the base case on this level of measurement noise. In the corresponding simulation for the base case, the plant converged to a point that deviated from the plant optimum. It should be noted that this level of noise is quite high with $\pm 10\%$ deviation on the constraint g_2.

With the highest noise level in plot D, the scheme is not converging to the plant optimum. However, the values of the hyperparameters does not change drastically. The reason is likely due to the noise level being too high even for this scheme, so the GP has to increase the length scale of the constraints to a level where too much information about the correlation in the data is lost. As a result it cannot describe the correct constraints and optimizer cease to function.

Table 5.5: Final hyperparameters, RBF length scale, l, and white noise, σ_n^2, for the three GPs for the third version of MA with GP.

		Gaussian process hyperparameters					
Noise		$(GP)^{\Phi_p - \Phi}$		$(GP)^{g_{p,1}-g_1}$		$(GP)^{g_{p,2}-g_2}$	
Level	Value	l	σ_n^2	l	σ_n^2	l	σ_n^2
A	$8 \cdot 10^{-7}$	3.9	10^{-6}	2.04	10^{-6}	2.78	10^{-6}
B	$8 \cdot 10^{-4}$	7.01	$1.8 \cdot 10^{-3}$	1.67	0.0122	3.13	$1.97 \cdot 10^{-3}$
C	$8 \cdot 10^{-3}$	6.77	0.274	6.98	0.716	6.01	0.222
D	$1.6 \cdot 10^{-2}$	7.17	0.399	11.3	0.89	7.29	0.392

5.7 Possible Improvements

After studying the three different versions of the GP with MA, it is evident that the last version with two iid estimates at each iteration performs best, in terms of noise resistance. However, there are several options that can be investigated to increase the performance even more, both in regards to noise tolerance and convergence rate of the schemes. Further a few options will be discussed.

There are several options to eliminate the finite difference error in the GP schemes. First, the GPs can be altered to describe the plant-model gradient difference directly. In all versions the GP is trained on data that describes the cost and constraint plant-model differences itself. The gradients are calculated using finite differences on the resulting GPs. Since the GPs can be trained to express any function, it could be trained to find the gradient functions directly. Then the finite differences and the belonging error would be eliminated, which could lead to

more accurate calculations and possibly faster convergence. However, training the GPs to represent the gradients, would involve evaluation of plant and model gradients for the training data. This would introduce the challenge of estimating the plant gradients, which is costly since it requires plant perturbations for each input. Therefore, training the GPs to describe the plant-model gradient difference would not be beneficial.

Another and more promising option to remove the error of the finite differences in the GP schemes, is to use automatic or analytical derivatives. Since the GP mean is a function, it is not necessary with an approximation to calculate the derivatives. However, this also allow for the step lengths used in the finite difference calculation to be very small. A step length of $h = 10^{-8}$ is used, which means that the the error in the finite differences are proportional to 10^{-8}. Therefore, using analytic or automatic derivatives for the gradient calculation for the GPs would likely not increase the performance of the MA and GP noteworthy, but it can easily be implemented to increase the accuracy.

Instead of adding one extra iid estimate, one could try to add two in order to increase the performance. As observed for version 3 of the scheme, one additional iid estimate lead to faster convergence and higher noise tolerance. With this intuition, it could be expected that providing even one more iid estimate at each iteration, would lead to the same favourable outcome. In practice, this would imply staying longer at each iteration point $\mathbf{u}_k$ in order to detect enough information for three measurements, with three different values of noise. This way, the GP would receive 50% more training data, which could make a difference especially in the first iterations when the training data set is small. However, GPs can become badly conditioned when doing this, and could lead to failed fitting. One should be especially cautious about staying at a single point to get multiple measurements if the plant-model mismatch is transient. The reason is that then the measurements would lead to different values after some time, and the GP would try to capture the transience. Therefore, it would theoretically be better to use a design of experiment or Bayesian optimisation methods to select the additional point.

As an extension of adding extra estimates at each iteration, one could calculate average measurements. This would improve the performance for the standard MA as well, since averaging can remove a lot of measurement noise. If for example 10 random samples are evaluated at each iteration, and then the average measurements are calculated, the average value would be expected to be closer to the noise free evaluations than most of the individual measurements. This could also improve the performance of the MA with GP, but the average data removes one important trait of the GP. By providing average data to the GP, it would not be possible to get a direct measure of the standard deviation in the data from the GP. This is an important feature. Although the standard deviation is not used here it is valuable because it provides a direct measurement of variations in the dataset. This could be compared with the white noise to reveal if the GP represents the input-output relationship of the data efficiently. Further, it has been observed that the GP handles the noise well. The aim of this implementation is to "decrease" the noise. Since it would limit the GPs ability to capture the uncertainty, it would not necessarily be favourable in the MA and GP case if the magnitude of the noise is not too large. Therefore it this option would be more relevant to implement in the case of standard MA.

The kernel determines the properties of a GP model and can influence the efficiency of the GP. One could try to find a better kernel than the one used. Among pre-defined kernels, there are options that describe for example linear and periodic relations. Since the functions we try describe are nonlinear and non-periodic, neither of these options would be relevant. However, there are many other pre-defined kernels and combination of kernels that could be used. Choosing or building a specific kernel can be difficult, and just like the true parameters are unknown, the true kernel is not revealed from the data. To decide which kernel to use, one could try out different kernels and compare the value of their marginal likelihood when used on the data[13]. Yet, for this case study it is unlikely that changing the kernel would affect the

results significantly. The reason is that the RBF kernel is universal and able to describe many models, and combined with the white noise kernel we get the desired property of describing some data variations as noise.

Finally, further work to improve the quality of the results would involve running all of the schemes with measurement noise multiple times to improve comparability. As previously stated, all of the "noisy" results displayed are from a single random run using the relevant settings. Because the noise is random, the results can be viewed as a snapshot of an infinite number of events. As a result, it is possible that some of the schemes that were observed to converge to the optimal solution will not be able to do so for every run. From this perspective, the results could be argued to be unrepresentative of the specific cases. However they show examples of what can happen for each case, and the changes in behavior trends from one case to another can be discussed. For example, it is likely that a scheme that converges quickly to the true optimum would do that for any run, but the iteration trajectories could vary. The issue is for schemes that are on the verge of convergence. For example somewhere between noise level C and D in Figure 5.9 the scheme fails to converge the optimum. With the results presented, it is impossible to say if the "convergence limit" is closer to noise level C or D. To be able study this further, many repeating runs at every noise level should be executed. By averaging the data in terms of for example final point and number of iterations before converging, one can secure that the results represent close to the mean of the current case. Then the results are better qualified for direct comparison of the values between the cases and not just the change in trends.

6 Conclusion

The results from Williams-Otto reactor simulation of standard MA, proves that MA solves the RTO challenge of dealing with structural plant-model mismatch. However, the standard MA struggles in presence of measurement noise. From the results it was observed that the MA did not converge even with small noise levels. With 0.001% noise on g_2, the optimizer ceases to function. The performance could be improved for noise level $2 \cdot 10^{-8}$ by increasing the filter parameters to 0.8 for both the input and the modifiers. Then the convergence rate was decreased and results in a slower optimizer. However, heavy filtering would unlikely be effective for higher noise levels. Therefore, use of standard MA as RTO approach in a chemical plant would require measurement devices with very high accuracy and possibly frequent calibrations in order to be useful.

MA with GP implemented to represent the plant-model mismatch in the constraint and cost functions, also converge to the plant optimum and tackles the structural plant-model mismatch in the case of no noise. The base case where measurement noise was introduced, the results indicate that MA with GP handles a noise level at least 4000 times larger than the standard MA. The optimizer converges for noise level $8 \cdot 10^{-4}$ which represents 1% noise on g_2. Thus, the MA with GP implementation clearly outperforms the standard MA in terms of noise tolerance. The reason is due to the GPs ability to represent a ratio of the variations in the data as noise. For the second GP version, an extra perturbed point was added at each iteration using 0.95*current point, resulting in a scheme converging to a point that was not the plant optimum. An explanation could be that the factor 0.95 was arbitrarily chosen, and in version three it was observed that providing two iid estimates of the same point would result in better performance.

The third MA with GP version indicated an ability to tolerate a noise level at least 40 000 times larger than the standard MA. The noise level corresponds to 10% measurement noise. This allows for very inaccurate measurement and usually one could expect devices to have higher precision than 10%. From this point of view the implementation of the third version could be excessive, if the measurements have a higher precision level. In addition, a drawback of this implementation is that it requires longer time at steady state in each RTO iteration compared to the base case.

In conclusion, in a perfect world with no noise, standard MA would be sufficient for optimization purposes in RTO when there is structural plant-model mismatch. However, its high sensitivity to noise may cause it to break if the noise is too large. The results have proven that for systems with significant measurement noise, a GP based MA could be beneficial due to the higher noise level tolerance. If the noise is large, version three of the MA with GP could be implemented, but that implies a tradeoff between steady state time and noise level tolerance.

References

[1] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. D. III, *Process Dynamics and Control*, 4th ed. Wiley, 2016, ch. 19.

[2] J. Matias and J. Jäschke, "Lecture Model Predictive control Module - Real time optimization and related challenges."

[3] C. Rasmussen and C. K. I. Williams, "Gaussian processes in machine learning," 2004.

[4] A. Marchetti, B. Chachaut, and D. Bonvin, "Modifier-adaptation methodology for real-time optimization," *Industrial amp; Engineering Chemistry Research*, vol. 48, no. 13, pp. 6022–6033, 2009. [Online]. Available: http://infoscience.epfl.ch/record/128111

[5] A. G. Marchetti, G. François, T. Faulwasser, and D. Bonvin, "Modifier adaptation for real-time optimization—methods and applications," *Processes*, vol. 4, no. 4, 2016. [Online]. Available: https://www.mdpi.com/2227-9717/4/4/55

[6] A. Marchetti, "Modifier-adaptation methodology for real-time optimization," pp. 25–42, 01 2009.

[7] T. d. A. Ferreira, H. A. Shukla, T. Faulwasser, C. N. Jones, and D. Bonvin, "Real-time optimization of uncertain process systems via modifier adaptation and gaussian processes," in *2018 European Control Conference (ECC)*, 2018, pp. 465–470.

[8] A. Scannell, "Gaussian Process Regression," https://www.aidanscannell.com/post/gaussian-process-regression/, 2019, read 23.10.2021

[9] L. F. Bernardino and S. Skogestad, "Lecture 7: Advanced Process control Module - Real-time optimization."

[10] A. Marchetti, "Modifier-adaptation methodology for real-time optimization," pp. 10–11, 01 2009.

[11] H. Sit, "Quick Start to Gaussian Process Regression," https://towardsdatascience.com/quick-start-to-gaussian-process-regression-36d838810319, 2019, accessed 27.09.2021

[12] M. Y. Walpole, Myers, *Probability and Statistics for Engineers and Scientists*, 9th ed. Pearson Education, Inc, 2011.

[13] D. Duvenaud, "Automatic model construction with gaussian processes," pp. 1–7, 2014.

[14] T. J. Williams and R. E. Otto, "A generalized chemical processing model for the investigation of computer control," *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, vol. 79, no. 5, pp. 458–473, 1960.

[15] Y. Zhang and J. Forbes, "Extended design cost: A performance criterion for real-time optimization systems," *Computers Chemical Engineering*, vol. 24, pp. 1829–1841, 09 2000.

[16] S. K. Morten Helbæk, *Fysikalsk kjemi.* Fagbokforlaget, 2006.

[17] J. F. Forbes, "Model structure and adjustable parameter selection for operations optimization," p. 149, 1994.

[18] D. Pollard, "Variances and covariances," http://www.stat.yale.edu/~pollard/Courses/241.fall2014/notes2014/Variance.pdf, 2014, chapter 4, accessed 29.10.2021

[19] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[20] A. Wächter and L. Biegler., "Scikit-learn: Machine learning in Python," *Mathematical Programming*, 2006.

[21] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] "The finite difference method," https://www.ljll.math.upmc.fr/frey/cours/UdC/ma691/ma691_ch6.pdf, accessed 10.11.2021

[24] E. W. Weisstein, "Taylor Series.From MathWorld–A Wolfram Web Resource," https://mathworld.wolfram.com/TaylorSeries.html, accessed 29.10.2021

A Gradient Approximation - Finite Differences

Finite differences is one of the least complicated and oldest method to estimate gradients and solve differential equations [23].

The derivative of a function $f(x)$ at a certain point x with a perturbation in x equal to h, can be defined as the following,

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{A.1}$$

It can be proven that when h approaches 0, the quotient becomes a good approximation of the gradient, hence the limit. Thus an approximation of the gradient can be expressed as

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \tag{A.2}$$

where h is constant. For smaller h, the accuracy of the approximation is increased. There are several possible expressions for finite difference gradient approximation. Equation A.2 is called forward difference, since the perturbation is in positive x-direction. Other examples are backwards and central difference.

The finite difference approximation error can be derived from the Taylor series, by neglecting higher order terms [24]. For a case where we want to estimate the gradient of a function $f(x)$ with respect to x, the Taylor series can be expressed as:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \cdots \tag{A.3}$$

The $\cdots$ captures all higher order terms. By altering the equation and isolation the first order derivative and divide with h, the following expression can be revealed

$$hf'(x) = f(x+h) - f(x) - \frac{h^2}{2!} f''(x+h_1) + \cdots \tag{A.4}$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2!} f''(x) + \cdots \tag{A.5}$$

Compared to the expression in Equation A.2, it is evident that they are equal if the higher order terms are neglected. By neglecting this terms, the magnitude of the error is found to be $O(h)$. The resulting gradient approximation with error term can be written as

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h). \tag{A.6}$$

Note that the error is proportional to h and therefore the approximation is referred to as a first-order approximation [23].

B Algorithms

B.1 Standard Modifier Adaptation

A pseudo code is presented to describe the algorithm that is used to simulate the standard MA for optimizing the Williams-Otto reactor. It consists of 9 main steps. For the case with no measurement noise, σ is set to 0.

Algorithm 1 Standard Modifier Adaptation

Step 0 - Initialization
Set initial u_0
Set filter parameters for the input $k_1 = k_2$ and for the modifiers $a_i = b_i = c, i \in 1, 2$
Set number of iterations n_k
Set initial modifiers $\epsilon_{i,0} = 0$, $\boldsymbol{\lambda}_0^{g_i} = \boldsymbol{\lambda}_0^{\Phi} = \mathbf{0}$
Set the noise level for the measurements σ
Calculate the cost noise σ_{Φ} from Equation 5.1

Step 1 - Initial optimization
Calculate u_0^* by optimizing the process model in Equation 4.13 and the initial modifiers from Equation 3.15

for $k = 0$ to $k = n_k$ **do**
Step 2 - Plant evaluation
Evaluate plant (Equation 4.12) and process model (Equation 4.13) at u_k to obtain:
Measurements and model states, X_p and X
Cost and constraints, Φ_p, $g_{p,1}$, $g_{p,2}$, Φ, g_1 and g_2 and implement random noise

Step 3 - Gradient calculation
Calculate plant gradients of cost and constraints with finite differences (Appendix A)
Calculate model gradients of cost and constraints by differentiation

Step 4 - Modifier calculation
Calculate the modifiers $\epsilon_{i,k}, \boldsymbol{\lambda}_k^{g_i}, \boldsymbol{\lambda}_k^{\Phi}$ from Equation 3.11, 3.13 and 3.14.

Step 5 - Filter modifiers
Filter $\epsilon_{i,k}, \boldsymbol{\lambda}_k^{G_i}, \boldsymbol{\lambda}_k^{\Phi}$ using Equation 3.17, 3.18 and 3.19

Step 6 - Optimization
Optimize Equation 3.15 s.t. 4.14 and Equation 4.13 with the calculated modifiers to obtain u_{k+1}^*

Step 7 - Filter input
Filter u_{k+1}^* using Equation 3.16

Step 8 - Convergence criteria
if $|u_{k+1}^* - u_k^*| < 10^{-8}$ **then**
Break for loop
end if
end for

B.2 Modifier Adaptation with Gaussian Processes

A pseudo code is presented to describe the algorithm that is used to simulate the base case of MA with GP for optimizing the Williams-Otto reactor. It consists of 9 main steps. For the case with no measurement noise, σ is set to 0. Compared with Algorithm 1, gradient calculation is substituted with update of training data and prediction with GP, and the modifiers are not filtered. Implementation of version 2 and 3 of the MA with GP(Section 5.6.2 and 5.6.3) require an additional plant evaluation step.

Algorithm 2 Modifier Adaptation with Gaussian Processes

Step 0 - Initialization
Set initial $u_0 = u_0^*$, filter parameters for inputs $k_1 = k_2$ and number of iterations n_k
Set initial modifiers $\epsilon_{i,0} = 0$, $\boldsymbol{\lambda}_0^{g_i} = \boldsymbol{\lambda}_0^{\Phi} = \mathbf{0}$
Set the measurement noise level σ and calculate the cost noise σ_{Φ} from Equation 5.1
Initialize the training input vector with 4 surrounding points of u_0
Initialize the 3 training output vectors, one for each f (Equation 3.20)

Step 1 - Initial training data evaluation
Calculate the outputs, f, for each training input points by evaluation of the plant (Equation 4.12) and process model (Equation 4.13) and add to the training output vectors

for $k = 0$ to $k = n_k$ **do**
Step 2 - Plant evaluation
Evaluate plant (Equation 4.12) and process model (Equation 4.13) at u_k to obtain:
Measurements and model states, X_p and X
Cost and constraints, Φ_p, $g_{p,1}$, $g_{p,2}$, Φ, g_1 and g_2 and implement random noise

Step 3 - Update of training data
Calculate the outputs, f in Equation 3.20, for the current point from the evaluated cost and constraints of the plant and process model and add to the training output vectors

Step 4 - Prediction with GP
Use the 3 training output vectors to fit 3 GPs and predict the output value at u_k^* of the three functions: $\Phi_p - \Phi, g_{p,1} - g_1, g_{p,2} - g_2$

Step 5 - Modifier calculation
Calculate the modifiers $\epsilon_{i,k}, \boldsymbol{\lambda}_k^{g_i}, \boldsymbol{\lambda}_k^{\Phi}$ from Equation 3.23, 3.24 and 3.22 using finite differences (Appendix A)

Step 6 - Optimization
Optimize Equation 3.21 s.t. Equation 4.14 and 4.13 with the calculated modifiers to obtain u_{k+1}^*

Step 7 - Filter input
Filter u_{k+1}^* using Equation 3.16

Step 8 - Convergence criteria
if $|u_{k+1}^* - u_k^*| < 10^{-8}$ **then**
Break for loop
end if
end for