1 2 9 0

UNIVERSIDADE D
COIMBRA

João Pedro Leonor Coutinho

SURROGATE MODELS FOR PROCESS AND CONTROLLER OPTIMIZATION

Master's Dissertation in the field of Chemical Engineering in the area of Process, Environment and Energy, supervised by Professor Marco Paulo Seabra dos Reis, PhD. and Professor Lino de Oliveira Santos, PhD., submitted to the Department of Chemical Engineering, Faculty of Science and Technology, University of Coimbra

December 2020

João Pedro Leonor Coutinho

SURROGATE MODELS FOR PROCESS AND CONTROLLER OPTIMIZATION

1 2 9 0 UNIVERSIDADE D COIMBRA

João Pedro Leonor Coutinho

Surrogate Models for Process and Controller Optimization

Master's dissertation in the field of Chemical Engineering in the area of Process, Environment and Energy, supervised by Professor Marco Paulo Seabra dos Reis, PhD. and Professor Lino de Oliveira Santos, PhD. and submitted to the Department of Chemical Engineering, Faculty of Science and Technology, University of Coimbra

Supervisors:
Professor Marco Paulo Seabra dos Reis, PhD
and
Professor Lino de Oliveira Santos, PhD

Institution:
Departament of Chemical Engineering
Faculty of Sciences and Technology of the University of Coimbra

Coimbra, December 2020

Em homenagem à minha madrinha (in memorium),
Isabel Maria Pisco Ribeiro

"When there is preparation, there is no fear"
Hwang Kee

Acknowledgments/ Agradecimentos

A realização desta tese, bem como do meu percurso académico, não seria possível sem o apoio ou ajuda de várias pessoas.

Em primeiro lugar, um obrigado à minha família, sem a qual eu não estaria neste momento a escrever estas palavras. Ao meu pai, Joaquim, pelos sacrifícios que fez e continua a fazer para que eu tenha a oportunidade de conseguir uma educação, bem como à sua prontidão em sempre responder a todas as perguntas começadas por "Porquê" de uma criança curiosa. À minha mãe, Maria José, pelos seus sacrifícios e por me pôr sempre em primeiro lugar. À minha avó Margarida, "Licas", pelo seu constante apoio e por sempre acreditar em mim e nas minhas capacidades. Ao meu avô José pela sua disponibilidade constante em me ajudar.

À minha namorada, Jéssica, por me ter aturado durantes constantes devaneios sobre modelos, controlo e otimização, bem como da vida em geral. Sem o seu constante apoio, nos bons e maus momentos, estas páginas estariam em branco.

Um obrigado aos meus orientadores que me deram a oportunidade de aprofundar conhecimento, bem como descobrir novos interesses, na área de modelação, otimização e controlo de processos. Ao Prof. Lino Santos pela sua constante disponibilidade em qualquer situação, bem como da sua paixão pelo ensino e pelos seus alunos. Se não fossem as suas aulas de TAMS e SP, provavelmente não teria desenvolvido o interesse que tenho por modelação e controlo de processos. Ao Prof. Marco Reis pela sua constante ajuda e por me ter introduzido a uma área que desconhecia, mas que cada vez mais gosto de descobrir. A sua vontade de inovação e de aprender é verdadeiramente uma inspiração.

Aos meus amigos de Coimbra e companheiros de várias noites atribuladas na Cidade dos Estudantes. Por ordem alfabética, para não ofender ninguém: Cláudia Gaspar, Daniela Pereira, Diogo Guerreiro, Diogo Salgueiro, Johnny Baptista, José Ascensão, Mariana Carvalho, Miguel Carmelo, Paula Oliveira, Ricardo Gameiro, Ricardo Graça, Silvino (mesmo longe, espero por ti no palco RUC), Telma Vaz. Por último, mas não menos importante, Hugo Ferreira, Vítor Sousa e Rodrigo Paredes. O que a receita uniu, ninguém separará. Também aos Engs. Eugeniu Strelet e Joel Sansana, pelas conversas sempre interessantes sobre modelação. Aos restantes que me tenha esquecido, fica devido um fino .

Ao Prof. Paulo Ferreira pelos conhecimentos transmitidos, não só em Fenómenos de Transferência e Papel, mas também na resolução de problemas, que ajudaram a formar a base

da engenharia que espero continuar a desenvolver. Ao Prof. Fernando Bernardo pelos conhecimentos transmitidos na área de otimização, que permitiram que pudesse compreender os conceitos aqui explorados.

Ao meu Mestre Pedro Dias, um obrigado pelos seus ensinamentos nas aulas de Karate, que mais tarde vim a descobrir que se aplicavam à vida em geral.

Obrigado,

João Pedro Leonor Coutinho

Resumo

Desenvolvimentos recentes em ferramentas de simulação de processos químicos baseadas em modelos complexos de primeiros princípios, têm permitido a realização de experiências simuladas avançadas, no âmbito de projeto, otimização e controlo de processos. Estas facilitam a tomada de decisões nestes domínios e permitem a redução da quantidade necessária de experiências físicas no processo, que muitas vezes são impraticáveis pela interferência com a sua operação normal e segura. Deste modo, experiências simuladas permitem elevadas poupanças económicas e ambientais, aumentando a segurança de bens e pessoas. No entanto, apesar do aumento de recursos computacionais, estas simulações continuam a ser morosas. Além disso, os modelos subjacentes a estas podem em alguns casos ser considerados como uma caixa-negra, onde apenas existe acesso a informação de entrada e saída. Estes fatores dificultam o uso destas ferramentas para tarefas como análise de sensibilidade ou otimização. Modelos substitutos baseados em técnicas de aprendizagem de máquina podem oferecer uma aproximação computacionalmente mais leve do modelo subjacente com base em dados de entrada e saída. Nesta dissertação, explora-se o uso de modelos substitutos para otimização global em duas aplicações relevantes na indústria, Otimização em Tempo Real (OTR) e sintonização de controladores Proporcional-Integral-Derivativo (PID).

Na primeira aplicação, foram comparados métodos de Otimização com Base em Modelos Substitutos (OBMS) com e sem uso de derivadas para OTR do reator químico Williams-Otto. Duas abordagens de otimização baseadas em amostragem sequencial, a Otimização Bayesiana (OB) e a Resposta de Superfície Estocástica (RSE), foram comparadas com o uso de diferentes tipos de modelos substitutos e planos de amostragem estáticos. Após a seleção, validação e otimização dos modelos susbtitutos, todos eles, tal como no caso da OB, permitiram encontrar o ponto ótimo de operação. No entanto, OB provou ser mais eficiente no que toca ao número de avaliações da função objectivo, evitando uma sobre-amostragem e fase extensa de seleção e validação de modelos.

Na segunda aplicação, foi proposta uma nova metodologia baseada em métodos de idenficação de sistemas e em OBMS multi-objectivo para sintonização de controladores PID. Esta metodologia é recomendada para ambientes de simulação, permitindo a identificação de limites dos parâmetros do controlador e diferenças de escala entre os diferentes objectivos do problema de otimização. Desta forma, reduz-se o esforço de tentativa e erro na otimização dos

parâmetros do controlador. Esta metodologia, em conjunto com a OBMS, constitui uma abordagem eficiente para sintonização controladores PID quer para modelos que requerem simulações numéricas morosas ou modelos de caixa-negra. Neste estudo, um modelo dinâmico de elevada ordem e altamente não linear é usado para comparar a OB e a RSE com abordagens de afinação convencionais em dois controladores PID diferentes. No geral, OB obteve o melhor desempenho. Como esperado de uma abordagem de otimização, este método de sintonização levou a uma melhoria do desempenho do controlador quando comparado com metodologias tradicionais. No entanto, é de salientar que a metodologia proposta é simples e rápida de usar, reduzindo a quantidade de tentativa e erro necessária. Testes de simulação em ciclo fechado demonstraram uma robustez do controlador satisfatória para cenários com alterações nos parâmetros do modelo e perturbações inesperadas. No entanto, o mesmo não se verificou para mudanças nas condicões operatórias.

Palavras-chave: *Modelos substitutos; Otimização de problemas de caixa negra; Otimização e Controlo de processos; Sintonização de controladores PID; Otimização Bayesiana; Resposta de Superfície Estocástica*

Abstract

Recent developments in chemical process simulation tools that based on complex first principle models, have enabled the use of more advanced simulated experiments in the fields of process design, optimization and control. These aid the decision making process in these domains and can reduce the amount of physical experiments on the process, which many times are infeasible because they upset normal and safe operation. This way, simulated experiments enable lower economic and environmental costs, increasing people and asset safety. However, despite the advances of computational resources, these simulations remain time consuming. Moreover, their underlying models can sometimes be considered a black-box, with access to only input and output information. Both these factors difficult their use for tasks such as sensitivity analysis or optimization. Machine learning based surrogate models can provide a computationally cheaper approximation of the underlying simulation model based on input and output data. In this dissertation, the use of surrogate models for global optimization is applied to two industrially relevant applications concerning Real Time Optimization (RTO) and Proportional-Integral-Derivative (PID) controller tuning.

In the first application, surrogate-based optimization (SBO) was compared with derivative-based and derivative-free optimization algorithms for RTO of the Williams-Otto chemical reactor. Two approaches based on adaptive sampling, Bayesian Optimization (BO) and Metric Stochastic Response Surface (MSRS) were compared with different model types and static sampling plans. After model selection, validation and optimization, all surrogate models, as well as BO, allowed finding the optimum operating condition. However, BO proved to be more efficient in terms of objective function evaluations, avoiding oversampling and an extensive phase of model selection and validation.

In the second application, a new methodology relying on system identification and multi-objective SBO was proposed for PID controller tuning. This methodology is applicable for simulation environments, and enables the identification of bounds on the controller parameters and scaling differences between multiple objectives in the optimization problem. This way, the effort associated with trial and error in the optimization of the controller parameters is reduced. In conjunction with surrogate models, this methodology provides an efficient approach for tuning PID controllers for models that require time consuming numerical simulations or black-box models. In this study, a high order and highly non-linear model is used to compare

BO and MSRS to standard tuning methods for two different PID controllers. Overall, BO displayed better performance. Although expected of an optimization-based tuning approach, this was accomplished in a highly automatic fashion, being simple and easy to use. Closed-loop simulations tests demonstrated a satisfactory controller robustness for scenarios with changing model parameters and unexpected disturbances. However, the same was not verified for changes in operating conditions.

Keywords: *Surrogate models; Black-box optimization; Process optimization and control; PID controller tuning; Bayesian Optimization; Metric Stochastic Response Surface*

Contents

Acronyms

ALAMO Automatic Learning of Algebraic Models for Optimization. 10
ANN Artificial Neural Networks. 13, 15–17, 20, 39, 42, 43, 45, 46, 48, 96
ARD Automatic Relevance Determination. 14

BO Bayesian Optimization. xvi, 24, 26, 27, 29, 34–38, 47, 48, 59, 61, 62, 66–74
BOBYQA Bound Optimization BY Quadratic Approximation. 38

CDF Cumulative Distribution Function. 25
CFD Computational Fluid Dynamics. 1, 2
CSTR Continuously Stirred Tank Reactor. 41
CV Cross Validation. 15, 44

DACE Design and Analysis of Computer Experiments. 4, 5
DF Describing Function. 63
DFO Derivative Free Optimization. 21, 22, 34, 38, 46, 49, 53, 71, 74
DoE Design of Experiments. 7, 27, 39, 43, 56, 57
DS Direct Synthesis. 32

EGO Efficient Global Optimization. 23, 24, 26, 34, 38
EHVI Expected Hypervolume Improvement. 34, 38
EI Expected Improvement. 24–26, 47, 59, 62, 67, 72

FOPTD First Order Plus Time Delay. 63

GA Genetic Algorithm. 22, 26, 29, 34, 36, 37, 46, 47, 59, 61, 65, 66
GLIS GLobal minimum using Inverse distance weighting and Surrogate radial basis functions. 23, 37
GP Gaussian Process. 13–15, 35

IAE Integral of the Absolute Error. 59, 66
IDW Inverse Distance Weighting. 24
IMC Internal Model Control. 32, 54, 63

IPTD Integrator-plus-time-delay. 57

ISE Integral of the Squared Error. xvi, 52, 63, 64, 66, 69, 70

ITAE Integral of the Time-weighted Absolute Error. 34, 57, 61, 68

KKT Karush-Khun-Tucker. 22, 23

LCB Lower Confidence Bound. 24, 25, 62, 67

LHD Latin-Hypercube Design. xv, 9, 10, 12, 38, 43, 44

LOLA Local linear approximation. 10

LQR Linear Quadratic Regulator. 36

MC Monte Carlo. 8–10, 17, 23, 26

MD Molecular Dynamics. 1, 2

MIMO Multiple Input Multiple Output. 16

MLE Maximum Likelihood Estimate. 13, 15, 94

MPC Model Predictive Control. 3, 31, 36–39, 73

MSE Mean Squared Error. 17

MSRS Metric Stochastic Response Surface. 24, 26–29, 47, 59, 61, 62, 65–67, 72, 73

NLP Non-linear Programming. 38, 42

NMPC Non-linear Model Predictive Control. 36, 38, 39, 73

ODE Ordinary Differential Equation. 38, 49, 54, 90, 92

OK Ordinary Kriging. 14, 39, 45

OLS Ordinary Least Squares. 13, 45

P Proportional. 32, 34

PDE Partial Differential Equation. 49, 90

PDF Probability Density Function. 25

PI Proportional Integral. 34, 35, 56, 57, 72

PI&D Piping and Instrumentation Diagram. xvi, 49, 89

PID Proportional integral derivative. 4, 31–34, 36, 37, 49–51, 54, 63, 68, 70, 72–74

PLS Partial Least Squares. 6, 10

POI Probability Of Improvement. 24, 25, 67

PSO Particle Swarm Optimization. 22, 29, 34, 36

QRLD Quasi-Random Low Discrepancy. 8, 10, 28

RBF Radial Basis Function. 4, 13, 17–19, 23, 24, 26, 27, 43, 45, 47, 97

RL Rotstein-Lewin. 63, 64, 66, 68, 72

RMSE Root Mean Squared Error. 20, 44, 97

RTO Real Time Optimization. 3, 4, 41, 42, 45, 48, 49, 52, 71

SAL Safe Active Learning. 35

SBO Surrogate-Based Optimization. xv, xvii, 21, 23, 41, 42, 47–49, 54, 56, 57, 59, 61, 62, 65, 67, 68, 71–74

SMS-EGO S-metric Efficient Global Optimization. 34, 38

SMS-EMOA S-metric Evolutionary Multiobjective Optimization Algorithm. 38

SOC Self-Optimizing Control. 6

SQP Sequential Quadratic Programming. 38, 46

SSA Smart Sampling Algorithm. 10, 11

SVM Support Vector Machine. 18, 39

SVR Support Vector Regression. 13, 18–20, 39, 43, 45, 46, 98

T-L Tyreus-Luyben. 54, 61

UAV Unmanned Air Vehicle. 38

Z-N Ziegler-Nichols. 34, 54, 56, 61

Symbols

X Matrix of input variables. 15, 24, 27

X_* Matrix of input variables for test set. 15

$k(.,.)$ Covariance Function. 14

n Number of training samples. 8, 9, 14, 15, 17, 18

p Controller output. 50

x Vector of input variables or vector of design variables. 17, 22, 53

y_* Vector of output variables in testing test. 14, 15

A_1 Pre-exponential constant of reaction 1, $[s^{-1}kgA]$. 91

A_2 Pre-exponential constant of reaction 2, $[s^{-1}kgB]$. 91

A_3 Pre-exponential constant of reaction 3, $[s^{-1}kgC]$. 91

C_{pA} Specific heat capacity of compound A, $[J.s^{-1}.K^{-1}]$. 90, 91

C_{pB} Specific heat capacity of compound B, $[J.s^{-1}.K^{-1}]$. 90, 91

C_{pC} Specific heat capacity of compound C, $[J.s^{-1}.K^{-1}]$. 91

C_{pE} Specific heat capacity of compound E, $[J.s^{-1}.K^{-1}]$. 91

C_{pG} Specific heat capacity of compound G, $[J.s^{-1}.K^{-1}]$. 91

C_{pP} Specific heat capacity of compound P, $[J.s^{-1}.K^{-1}]$. 91

C_{pR} Specific heat capacity of reactor mixture, $[J.s^{-1}.K^{-1}]$. 89–91

C_{pW} Specific heat capacity of cooling water, $[J.s^{-1}.K^{-1}]$. 90, 91

C Trade-off constant for Support Vector Regression models. 19, 20, 98

D_C Internal coil diameter, [m]. 90, 91

D_R Reactor diameter, [m]. 90, 91

D Number of design variables. 8, 9, 12, 14, 22, 59, 61

Ea_1 Activation energy of reaction 1 divided by ideal gas constant, $[K]$. 91

Ea_2 Activation energy of reaction 2 divided by ideal gas constant, $[K]$. 91

Ea_3 Activation energy of reaction 3 divided by ideal gas constant, $[K]$. 91

F_A Reactant A feed stream mass flowrate, $[kg.s^{-1}]$. 41, 42, 48, 52, 90

F_B Reactant B feed stream mass flowrate, $[kg.s^{-1}]$. 41, 42, 48, 50, 52, 90

F_R Reactor exit stream, $[kg.s^{-1}]$. 41

I Improvement in objective function value. 25

K_D PID controller derivative gain. 50, 51, 53, 54, 65, 67

K_I PID controller integral gain. 50, 51, 53, 54, 59, 60, 62, 65, 67

K_P PID controller proportional gain. 50, 51, 53, 54, 59, 60, 62, 65, 67

K_c Controller Gain. 54, 58, 64

K_{cu} Ultimate Gain. 54, 58, 102

K Process gain. 57, 58, 63, 64, 102

L_C Internal coil length, [m]. 91

L_R Reactor liquid level, [m]. 50, 57, 90

L_{Rsp} Reactor liquid level setpoint, [m]. 52

P_u Ultimate Period, [s]. 54, 58, 102

Q_R Reactor exit stream volumetric flowrate, $[m^3.s^{-1}]$. 50, 57, 90

Q_{Rmax} Maximum reactor exit stream volumetric flowrate, $[m^3.s^{-1}]$. 93

Q_{Rmin} Minimum reactor exit stream volumetric flowrate, $[m^3.s^{-1}]$. 93

Q_{cinmax} Maximum cooling water volumetric flowrate, $[m^3.s^{-1}]$. 93

Q_{cinmin} Miminum cooling water volumetric flowrate, $[m^3.s^{-1}]$. 93

Q_{cin} Cooling water volumetric flowrate, $[m^3.s^{-1}]$. 50, 65, 90, 92

Q_{cool} Heat transfer rate between the reactor mixture and cooling water, $[W]$. 90–92

T_A Reactant A feed stream temperature, $[K]$. 68, 90

T_B Reactant B feed stream temperature, $[K]$. 68, 90

T_C Cooling water temperature, $[K]$. 91

T_R Reactor mixture temperature, $[K]$. 42, 48, 50, 90

T_{Cin} Cooling water feed temperature, $[K]$. 68, 90

T_{Cout} Cooling water exit temperature, $[K]$. 92

T_{Rsp} Reactor mixture temperature setpoint, $[K]$. 52

T Relay hysteresis. 54, 102

U Global heat transfer coefficient, $[W.m^{-2}.K^{-1}]$. 68–70, 90, 91

V_R Volume of reactor mixture, $[m^3]$. 91

V_T Total reactor volume, $[m^3]$. 91

V_{coil} Volume of internal coil, $[m^3]$. 91

X_A Mass fraction of compound A in reactor mixture. 90

X_B Mass fraction of compound B in reactor mixture. 90

X_C Mass fraction of compound C in reactor mixture. 90

X_E Mass fraction of compound E in reactor mixture. 42, 90

X_G Mass fraction of compound G in reactor mixture. 90

X_P Mass fraction of compound P in reactor mixture. 42, 90

ΔH_{R1} Heat of reaction 1, $[kJ.kg^{-1}]$. 90, 91

ΔH_{R2} Heat of reaction 2, $[kJ.kg^{-1}]$. 90, 91

ΔH_{R3} Heat of reaction 3, $[kJ.kg^{-1}]$. 90, 91

Δt PID controller sampling time, [s]. 50, 51, 53, 55, 57–60, 62, 63, 65, 67

Δz Length of discretization step, [m]. 90, 91

$\Phi(.)$ Chapter 2: Vandermonde matrix (Polynomial Regression), Interpolation matrix (Radial Basis Function). 13, 25

α Lagrange multiplier. 19

$\bar{Q_C}$ Bias value of cooling water feed stream volumetric flowrate, $[m^3.s^{-1}]$. 93

$\bar{Q_R}$ Bias value of reactor exit stream volumetric flowrate, $[m^3.s^{-1}]$. 93

β Vector of polynomial regression coefficients. 13, 14, 18

ϵ Support Vector Regresion approximation tolerance. 19, 20

γ Constant (Radial Basis Function), Gaussian kernel parameter (Support Vector Regression). 18, 19, 97, 98

$\hat{\mu}$ Predicted mean by Kriging model. 15, 25, 26

$\hat{\sigma}$ Predicted Standard Deviation by kriging model. 15, 25, 26

$\hat{y}$ Vector of predicted output variable values. 15, 16, 19

λ Radial Basis Function weight coefficient vector. 17, 18

μ Support Vector Regression bias. 19, 20

ω_u Ultimate frequency. 102

$\phi(.)$ Chapter 2: Activation function(Artificial Neural Networks), Radial Basis Function(RBF), Kernel(SVR); Chapter 3: Normal Probability Distribution Function. 25

$\psi(.)$ Support Vector Regression kernel. 19

ρ_R Reactor mixture volumic mass, $[kg.m^{-3}]$. 90, 91

ρ_W Cooling water volumic mass, $[kg.m^{-3}]$. 90, 91

σ_f Standard deviation of input signal. 14

τ_D Derivative time constant, [s]. 54, 64

τ_I Integral time constant, [s]. 54, 58, 64

τ_c Closed-loop time constant, [s]. 58, 60

τ Process time constant. 51, 63–65, 102

θ Vector of kriging model parameters (Chapter 2), Process time delay (Chapter 6). 15, 58, 60, 63, 64, 102

ζ Target in Probability Of Improvement acquisition function. 25

a Controlled variable oscillation amplitude. 54, 58

b_h Bias vector of hidden layer. 16

b_o Bias vector of output layer. 16

d Relay amplitude. 54, 102

e Error. 50, 52, 54, 102

f_{min} Current Best Objective function. 25

f Objective function. 21, 22, 24, 27–29, 53, 59, 65

g Optimization inequality constraints. 22

h Optimization equality constraints. 22

k_1 Rate of reaction 1, $[s^{-1}]$. 90

k_2 Rate of reaction 2, $[s^{-1}]$. 90

k_3 Rate of reaction 3, $[s^{-1}]$. 90

k Trade-off constant in LCB acquisition function. 25

l Length-scale. 14

m_R Reactor mass, [kg]. 90, 91

n_* Number of testing samples. 15, 20

u_{max} Maximum value of manipulated variable, or process input. 53

u_{min} Minimum value of manipulated variable, or process input. 53

u Manipulated variable, or process input. 50, 53, 54, 102

w Chapter 2: Neuron weight coefficient (Artificial Neural Networks), weight coefficient (Support Vector Regression); Chapters 3 and 6: Objective function weight vector. 16, 19, 29, 53, 59, 65

x^* Adpative sample, or incumbent. 24, 27

x_{lb} Lower bounds on decision variables. 22, 28, 53

x_{ub} Upper bounds on decision variables. 22, 28, 53

y_m Measured process output. 50

y_{sp} Controlled variable setpoint. 50, 52, 53

y Vector of output variables, process output (Chapter 6)). 14, 15, 18, 50, 52, 53, 102

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and scope of the thesis

Nowadays, global competition in the process industries motivates chemical manufacturing companies to design and operate production facilities with a tighter cost margin. Beyond revenue increase, process industries also face increasingly stringent regulations regarding environmental pollution, motivating more efficiently operated processes able to minimize waste production and meeting legal emission limits. This leads to considerable investment of time and resources to optimal process design, operation and control. Process modelling and optimization are indispensable tools to achieve these goals.

Models are abstract mathematical representations of a physical process and can be derived using knowledge based on first principle laws from physics or process data. In recent years, the advancement of computational resources and simulation tools based on complex first principle models, such as Computational Fluid Dynamics (CFD) or Molecular Dynamics (MD), have allowed engineers to use more advanced simulation-based experiments to gain knowledge and solve increasingly complex problems. These simulations can greatly reduce the number of physical experiments required, especially in early design phases, and therefore result in a reduction of costs, resources and time. However, in spite of advances in computing power and the use of distributed parallel computers, such complex simulations are still time consuming, mainly due to the solution of complex large scale systems of non-linear differential equation. Consequently, in some cases, a single simulation may require minutes, hours or even days, rendering tasks that require hundreds or thousands of simulations, such as optimization or Monte Carlo simulations, impossible or infeasible in practice without high computing resources. Moreover, in addition to being time consuming, the inner workings of the models used in these simulations are complex and usually not available to the user. In these cases, the simulator can be thought of as a black-box model, for which only input and output data is available. Explicit algebraic expressions for the model derivatives are also not readily available, and expensive to estimate

numerically, which renders traditional and well proven gradient based optimization strategies ineffective (Rios and Sahinidis (2012)). Besides simulators relying on CFD or MD, this is also the case for widely used commercial chemical process flowsheet simulators, like Aspen Plus® or Aspen HYSIS®. Despite being computationally less demanding, these simulators mainly use a sequential-modular structure calculation that leads to numerical noise and reduces finite difference accuracy (Caballero and Grossmann (2008)). This is especially problematic when heat or mass recycle streams are present (Straus (2018)). Due to this issue, optimization requires the use of specialized algorithms to recover from convergence failures (Cozad et al. (2014)).

Because of these difficulties, systematic use of complex first principle simulators for tasks such as sensitivity analysis, optimization or feasibility analysis is difficult, and the user is usually left to using intuition and highly specialized or time consuming approaches. A practical solution to avoid the mathematical complexity in simulation is to use surrogate models.

Surrogate models (Bhosekar and Ierapetritou (2018), McBride and Sundmacher (2019)), Response surface models or metamodels (Wang and Shan (2007)) provide an algebraic approximation of a complex system or model based on input-output data. The underlying model is treated like a black-box, assuming that little or no information about it is known. These surrogate models offer the advantage of mimicking the behaviour of the black box model, while being much faster to evaluate. Because of this, computationally cheaper surrogate models can be used in situations whenever the simulation of a black-box type model is necessary, including the use of a complex first principles based simulator or other simulation experiments that are computationally demanding and with unknown or mathematically intractable inner structure.

Surrogate models can mainly serve tasks that require model approximation, optimization, feasibility or sensitivity analysis. Therefore, they can be used in a wide range of fields in process systems engineering including process and product design, optimization, control, planning and scheduling and supply chain management. A few recent applications that require time consuming simulations include chemical reactor operational optimization (Yang et al. (2018), Kong et al. (2020)) and design (Park et al. (2018)) through CFD and estimating molecular properties (Kadupitiya et al. (2020)) or nanowire design (Mukhopadhyay et al. (2016)) using MD simulations. In separation processes, surrogate models have been used for global optimization of distillation columns with highly non-ideal mixtures (Keßler et al. (2019)), replacing computationally expensive thermodynamic models for phase equilibrium calculations (Nentwich and Engell (2019)) and optimization of a CO2 adsorption process under uncertainty using an expensive simulation model (Hüllen et al. (2020)). Beyond time consuming first principles models, surrogate models have also been recently used to tune the hyper-parameters of the data-driven part of a hybrid model (Bikmukhametov and Jäschke (2020)). A plethora of references to the application of surrogate modelling in chemical engineering is available in recent surveys (Kajero et al. (2017), McBride and Sundmacher (2019)).

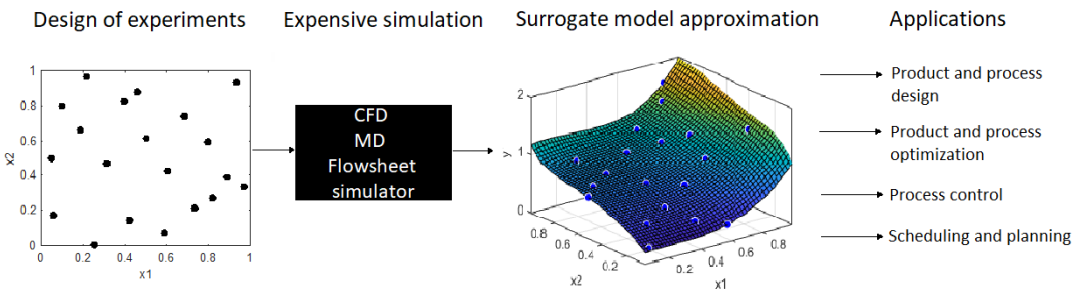


Figure 1.1: Representation of the surrogate model generation and applications

Industrial processes require that operating conditions are maintained within a given range of values, either to maximize revenue, for safety reasons or to comply with environmental regulations. This can only be achieved using process control, through the use of controllers that are designed to maintain some process variables at their desired values by manipulating others. A simplified representation of the classic hierarchical structure for process control is presented in Figure 1.2 (Seborg et al. (2017)).

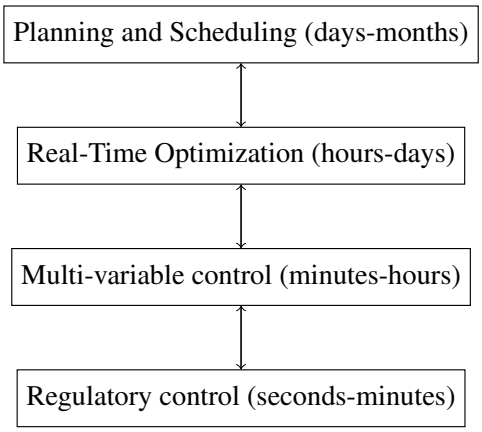


Figure 1.2: Classic process control hierarchy

The top layer is where medium and long term decisions regarding production planning are made. In the Real Time Optimization (RTO) layer, a detailed plant model, usually steady-state, is used to find the economically optimum operating points whenever changes in product prices, raw material costs and process disturbances occur. The optimum operating points are defined as the setpoints for the subsequent control layers, which can include adavance multi-variable and constraint control strategies such as Model Predictive Control (MPC). At the bottom layer, standard or advanced feedback and feedforward control strategies are used to maintain controlled variables at their optimum setpoint, defined in the layers above. More information is available in Seborg et al. (2017). Surrogate models can be used to solve problems concerning model complexity or unavailable explicit equations across all these layers.

In this thesis, surrogate models are used for RTO when the plant model is a black-box and for controller tuning in the regulatory layer. Additional uses for other layers are also identified.

1.2 Thesis Objectives

The work developed in this thesis is guided by the following objectives:

- Introduce the concepts of surrogate modelling and surrogate-based optimization.
- Review applications of surrogate modelling in process control.
- Through a simple example compare the performance of different surrogate models, sampling plans and strategies for RTO where the process model is a black-box. Demonstrate the advantages of adaptive sampling strategies for surrogate-based optimization.
- Develop and apply a simple methodology to tune digital PID controllers in simulation scenarios using surrogate-based optimization. Demonstrate the advantages of this approach by comparison with traditional benchmark tuning methods.

1.3 Thesis Outline

This thesis is composed by 7 chapters, organized as follows:

- In chapter 2 the concept of Design and Analysis of Computer Experiments (DACE) is introduced, along with different sampling plans, surrogate models and validation techniques.
- Chapter 3 discuses the use of surrogate based optimization for global optimization of black-box problems. Different methods are reviewed, with an emphasis on those using global surrogate models, where two different approaches using Radial Basis Function (RBF) and kriging are introduced. In addition, strategies to deal with multiples objectives when using surrogate models are briefly reviewed.
- In chapter 4, the state-of-the-art on different applications for surrogate models in the field of process control is presented.
- Chapter 5 presents a simple case study regarding the use of surrogate models for RTO of a chemical reactor when the model is unknown. In this example, different surrogate models and sampling procedures are used. The performance of surrogate-based optimization is compared with traditional optimization methods.
- In chapter 6, a methodology using surrogate-based optimization is applied to PID controller tuning in simulation. A complex non-linear and high order dynamic model of the reactor that shows open-loop unstable behaviour is used to demonstrate the advantages of this approach when compared with traditional tuning approaches.
- Chapter 7 concludes the thesis work. Conclusions are summarized and opportunities for future work are identified.

Chapter 2

Design and Analysis of Computer Experiments

In the first section of this chapter the concept of surrogate modelling is introduced, followed by explanation of how to design experiments for surrogate models. Sections 3 presents the most popular types of surrogates models. In section 4, methods to validate such models are discussed.

2.1 Introduction to surrogate modelling

A black-box simulation refers to scenarios where the expressions of the underlying mathematical model are either unknown or too complex to be analysed explicitly. Figure 2.1 presents a representation of the information flow in these types of problems:

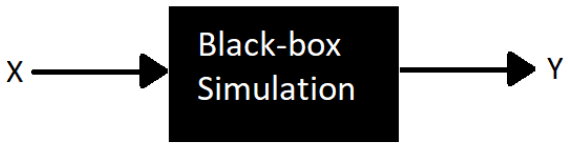


Figure 2.1: Black-box simulation

It is generally assumed that only information about the input, X, and output variables, Y, is known about these simulations, and that these are also time consuming. The systematic analysis of these problems is referred as Design and Analysis of Computer Experiments (DACE), a concept made popular by the article of Sacks et al. (1989). The main concepts of DACE are the experimental design and the use of response surface, or surrogate models. These are machine learning based models that are used to provide a computationally inexpensive approximation

of the mathematical relationship between the input and output variables. Surrogate modelling generally encompasses the following set of steps:

- Identify the design and output variables, as well as bounds on the design space
- Generate values for the design variables and simulate them on the black-box simulation
- Fit a surrogate model on the input and output data
- Validate the surrogate model

Depending on the application, modifications within each step or additional steps can be used.

2.2 Selecting variables and defining the design space

The first step in an experiment, whether physical or simulated, is to identify the variables of interest. This includes choosing the independent variables x, also known as input or design variables, that are going to be manipulated to cause an effect on other dependent, or output, variables y.

The selection of y should always be done with attention to the modelling objective. For instance, to gain system knowledge, the goal may be to approximate several independent variables. Instead, for optimization, the goal is to generally approximate an objective function, which can depend on these variables.

Several approaches have been taken to deal with the high dimensionality of the design variables. These achieve dimensionality reduction through the use of data-driven methods such as variable selection using regression methods (Bhosekar and Ierapetritou (2018), Kim and Boukouvala (2020)), Zhai and Boukouvala (2019)), projection of variables into a lower dimensional intrinsic space using non-linear manifolds (Lovelett et al. (2019)), Partial Least Squares (PLS) regression (Straus (2018)) or even using concepts from process control such as Self-Optimizing Control (SOC) (Straus and Skogestad (2018)), to name a few. Since the number of design variables considered in the case studies of this thesis is low, concepts and approaches to variable selection or reduction are not discussed in detail. Interested readers are referred to the aforementioned references.

After selecting the design variables, lower and upper bounds within which they can vary needs to be defined. This set of values may be referred to as the design space, and accurately defining it beforehand is of high importance, since reducing the design space can lower the problem complexity (Forrester and Keane (2009)). Domain specific knowledge should always be used to define both variables and variable bounds (Straus (2018), Piga et al. (2019)).

2.3 Design of experiments for surrogate modelling

After the selecting the design variables, the next step is the generation of input data points, or samples, within the design space to perform the experiments. Both quality and performance

of a surrogate model depend strongly on a sufficient number of quality samples used to build the model. This can be accomplished through a procedure to plan and define conditions for experimental trials known as Design of Experiments (DoE).

Classic DoE methods for physical experiments include designs such as the full and half factorial (Fisher (1926)), Box-Behnken (Box and Behnken (1960)), Central Composite and Plackett-Burman (Plackett and Burman (1946)). These methods are designed to be robust against the unavoidable variance due to the stochastic nature of physical experiments (Garud et al. (2017a)). On the other hand, deterministic computer simulations, without added noise, do not present experimental variance. In this regard, designs for simulated experiments diverges from classic DoE, since common concerns due to randomness of physical experiments, such as blocking and replications are not as important in the sampling procedure. Instead, the focus is to accomplish the modelling goals while requiring the lowest amount of samples, considering these usually require computationally demanding simulations. Sample generation can be divided into either static, or one-shot, sampling or adaptive, or sequential, sampling. A simplified flow diagram of both strategies is presented in Figure 2.2.

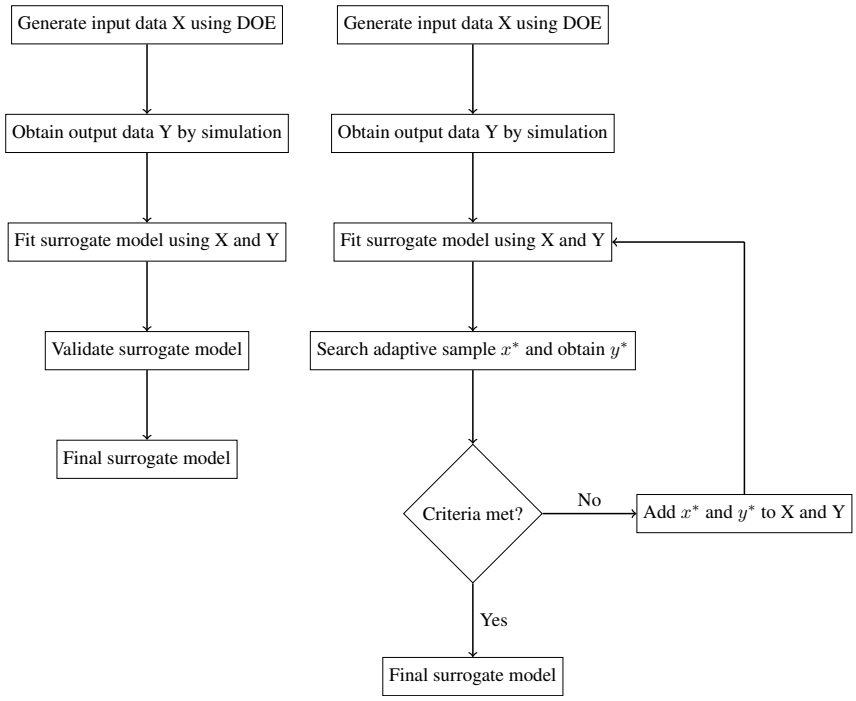


Figure 2.2: Flowchart of static (on the left) and adaptive sampling (on the right)

In static sampling, a sample of predetermined size is generated through DoE and the model is fitted to the data. In adaptive sampling, after the model is fitted, a new sample that enhances its quality is searched according to a criterion that can include information provided by the model. This sample is simulated to obtain the respective output data and added to the initial dataset. This sequential procedure is then repeated until a pre determined stopping criterion is met.

2.3.1 Static Sampling

Monte Carlo (MC) sampling was the first formal method for sample generation for computer based experiments and remains one of the most used methods to this day (Garud et al. (2017a)). The MC method was proposed by Metropolis and Ulam (1949) and it uses a pseudo-random number generator to generate sample points, with the hope that the inherent randomness results in space filling. The shortcoming of these designs is that a sample of finite size may lead to clustering and unrepresented regions of the design space, resulting in an ineffective space filling design (Garud et al. (2017a)).

Quasi-Monte Carlo sampling methods use a Quasi-Random Low Discrepancy (QRLD) sequences to generate samples. The quasi-random designation comes from the fact that these sequences are of deterministic nature and low discrepancy implies a close to uniform distribution of points in the design space. Several quasi-random designs exist such as Halton (Halton (1964)), Hammersley (Hammersley (1960)) and Sobol (Sobol (1967)). QRLD sequences use the concepts of inverse radix numbers and prime numbers. Although these sampling methods aim to achieve a space-filling design, they do not incorporate a formal quantification of space-filling during the sample generation procedure (Garud et al. (2017a)). The following designs consider space filling criteria explicitly to generate samples.

Intuitively, one would think to start with the simplest form of space filling, the classic full factorial design, where the samples are arranged as in a regular grid, such as the one in Figure 2.3.

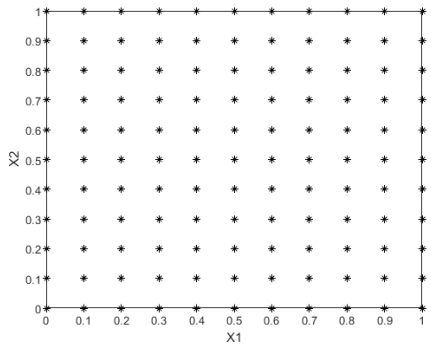


Figure 2.3: Full factorial design with 121 samples

The total number of samples required for this design increases exponentially with the number of design variables, D, in the fashion n^D, where n is the number of samples in each dimension. One can see how quickly this sampling design gets out of hand in terms of computational complexity, even for functions cheap to evaluate, as the number of dimensions increases. This is commonly known as the *curse of dimensionality* (Forrester et al. (2008)).

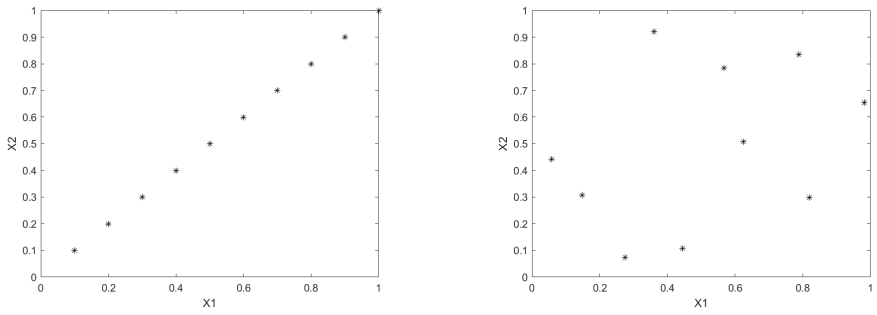
A number of space-filling criteria exist in the literature that can be divided into two general categories: uniformity-based and distance-based.

Uniformity-based criteria use a measure of discrepancy, which quantifies the departure of a given design from a uniform design. An example based on this criteria is the Uniform Design (Fang et al. (2000)), which places samples that minimize discrepancy.

Distance-based criteria, as the name implies, consider the distance between sample points to quantify space filling. Two examples of distance-based designs are Maximin and Minimax designs (Johnson et al. (1990)). Maximin designs generate samples such that the maximin criteria, the minimum distance between any two points, is maximized. On the other hand, Minimax design minimizes the maximin criteria.

For further details on the mentioned space-filling as well as other geometric criteria, such as Voronoi tessalation or Delaunay triangulation, cf. Garud et al. (2017a) and the references mentioned therein.

Latin-Hypercube Design (LHD) was proposed by McKay et al. (1979) to overcome issues associated with MC sampling variations and is one of the most popular designs across various fields (Garud et al. (2017a)). LHD divides a design space with D dimensions into n bins, placing each sample into a bin such that no two samples are present in each row or column (in the case of a 2 dimensional design). However, because the LHD configuration and sample placement within the bins are random, this design may not result in adequate space-filling. A well known worst case example of this shortcoming is presented in Figure 2.4.



(a) **(b)**

Figure 2.4: LHD with 10 samples: a) worst case design; b) space-filling design.

The space-filling ability of LHD can be enhanced using optimal-LHD designs. These optimize the sample spatial arrangement in LHD according to a space-filling criterion, for example, maximin. The reader is referred to Garud et al. (2017a) for a review of different approaches. The main issue in these methods is that the optimization problem involved in the sample location selection is computationally intensive (Liu et al. (2018), Garud et al. (2017a)).

All static designs reviewed so far are generic in nature, focusing only on spatial distribution of the samples within the design space and ignoring the system under study. Designs that consider information about the system have also been used in the literature (Sacks et al. (1989)).

Because of different systems and surrogate models, no definitive conclusion exists about

the absolute best design. However, Garud et al. (2017a) compared the space filling abilities of MC, LHD, and QRLD designs across several dimensions, including high dimensional design spaces. Overall, a variation of the Sobol design showed the best performance.

2.3.2 Adaptive Sampling

In the previous section, the sampling procedures discussed generated all points at once, in a one-shot or static manner. Despite popularity, these methods may lead to under-sampling or oversampling (Crombecq et al. (2011), Garud et al. (2017a)). With under-sampling, the number of samples is insufficient to build an accurate surrogate model. On the other hand, oversampling may result in a waste of computational resources when the simulation output is too expensive to obtain. These problems led to the development of adaptive sampling methods, that aim to build an accurate surrogate model using only the necessary samples to do so (Liu et al. (2018)).

Two key concepts in adaptive sampling are design space exploration and exploitation. Exploration refers to placing sample points on unsampled regions of the design space where model uncertainty is high, in a space filling manner, similarly to the static designs. On the other hand, exploitation incorporates system information and aims to sample interesting, non-linear or complex, regions of the design space near existing sample points. Because exploitation takes different meanings for different goals, a preliminary clarification is important.

Surrogate models can generally be used for three distinct classes of problems including modelling, optimization and feasibility analysis (Bhosekar and Ierapetritou (2018)). When discussing adaptive sampling approaches, it is important to clarify what the purpose of building the surrogate model is because the adaptive search strategy is different for each situation. For instance, when global approximation is the goal, exploitation means placing samples where a measure of error is larger (e.g. Cozad et al. (2014)), while in the context of optimization, exploitation means placing samples near points with better objective function values.

2.3.2.1 Adaptive sampling for global modelling

Surrogate models using adaptive sampling can be used to approximate and gain knowledge of the underlying black-box model in the entire design space. Two types of strategies for these situations include the use of pure exploration or a balance of exploration and exploitation. While the former strategy may be employed, the latter has been subject to higher research focus, because sample placement is done in a more intelligent manner while avoiding the curse of dimensionality (Garud et al. (2017a)). A few notable examples include LOLA-Voronoi (Crombecq et al. (2009)) Automatic Learning of Algebraic Models for Optimization (ALAMO) (Cozad et al. (2014), Wilson and Sahinidis (2017)), Smart Sampling Algorithm (SSA) (Garud et al. (2017b), Garud et al. (2018)), and, more recently, an sampling procedure using Partial Least Squares (PLS) regression as a termination criterion (Straus (2018), Straus and Skogestad

(2019)). For a further discussion of adaptive sampling procedures for global approximation, the reader is referred to the reviews of Crombecq et al. (2011), Garud et al. (2017a) and Liu et al. (2018). Garud et al. (2018) also compared their SSA method against other popular adaptive sampling procedures.

2.3.2.2 Adaptive sampling for optimization

Instead of gaining system knowledge, the goal of using surrogate models can be simply to solve an optimization problem that requires an expensive simulation with unavailable derivative information (Bhosekar and Ierapetritou (2018)). In this case, a surrogate model can be created to approximate the relationship between the objective function and the decision variables. Two adaptive sampling strategies can be considered for this purpose (Wang and Shan (2007)):

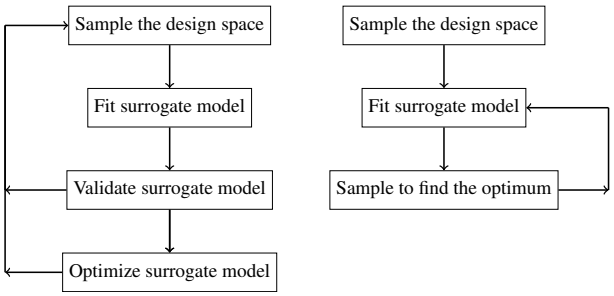


Figure 2.5: Adaptive sample approaches for optimization

The first strategy, on the left, includes surrogate model validation and optimization in the adaptive sampling procedure. Samples are searched to refine the surrogate model approximation, before it is optimized using an external optimization algorithm. The second strategy, on the right, consists on using adaptive sampling to directly search for the optimum value of the underlying function. In this approach, the demand on global model accuracy is lowered since it is not a criterion for searching adaptive samples (Wang and Shan (2007)).

One important caveat has to be made about the use of an adaptive sampling procedure for global approximation and further optimization. It is not guaranteed that a surrogate model that globally approximates the original design space can lead to the global optimum of the underlying function when used for optimization. This is because the surrogate model is unlikely to be sufficiently accurate, initially, in the global optimum region (Forrester et al. (2008)). Additionally, the optimization algorithm may exploit approximation errors of the surrogate model (Eason and Biegler (2016)). Also, this approach leads to a inefficient use of time and computational resources, since samples are wasted on modelling regions with no optima (Forrester et al. (2008), Liu et al. (2018)). However, the use of globally accurate surrogate models for optimization goals may still be adequate in the cases when these models are made to replace single units, or subprocesses, to be used in conjunction to solve a larger optimization problem (Straus (2018), Straus and Skogestad (2018)). Because large scale optimization is not considered in this

thesis, adaptive sampling methods for the purpose of optimization are preferred. Such methods, along with strategies to balance exploration with exploitation are discussed in Chapter 3.

2.3.2.3 Adaptive sampling for feasibility analysis

Another application surrogate models is to identify the conditions under which a process is feasible, ie. able to satisfy all constraints, when expensive black-box simulations are required (Bhosekar and Ierapetritou (2018), McBride and Sundmacher (2019)). In this problem, the surrogate is used to approximate the feasibility function, which indicates if constraints are violated, given a set of input parameters and output simulation data. Adaptive sampling can be used to reduce the amount of samples, which is usually high due to the presence of multiple constraints. In contrast to optimization, the objective is to define the boundary of the feasible space. For more information is referenced to Bhosekar and Ierapetritou (2018) and McBride and Sundmacher (2019), where references to different approaches are available.

2.3.2.4 Remarks on adaptive sampling difficulties

Adaptive sampling may appear as a definitive answer to sampling for surrogate model construction. However, their use for high dimensional problems remains a challenge and in some cases the adaptive search is still computationally expensive (Liu et al. (2018), Garud et al. (2017a)). Another disadvantage is that both the initial sample size and design are left to be chosen by the user and, to the best author's knowledge, no definitive conclusions exist in this regard.

Through a literature review, it seems that the most popular design is LHD, but in most cases, its use is not justified. The initial sample size is a compromise between sufficient space filling samples and a sufficiently large budget of adaptive samples. While a initial sample too small can lead to poor surrogate model approximation and thus, misguide the adaptive sample search (Liu et al. (2018)), a large number of initial samples may be a waste of space-filling function evaluations that could be better spent on adaptive samples. The adequate selection of the initial sample size depends on the problem dimensionality and complexity, computational budget and surrogate model characteristics. A set of empirical recommendations considering these factors is available in Liu et al. (2018). In this work, two heuristics regarding the number of design variables are considered: a initial sample size with $10D$ (Loeppky et al. (2009)) and with $5D$ samples (Liu et al. (2016)).

2.4 Surrogate models

After variable selection and the generation of samples, a surrogate model is built (or trained) on the set of input and output variable values, ir order to approximate the function be-

tween these two sets of variables. There is also the possibility of using a combination of multiple surrogate models (Bhosekar and Ierapetritou (2018)), but, for brevity sake, these methods will not be discussed in this work. A variety of machine learning models can be used as surrogates, from a simple linear regression model to a complex deep neural network. In this section the most commonly used models are discussed, including interpolation techniques, such as Kriging and Radial Basis Function (RBF), and regression techniques like polynomial regression, Artificial Neural Networks (ANN) and Support Vector Regression (SVR).

2.4.1 Polynomial Regression

Polynomial regression models are the simplest and perhaps the most widely used surrogate model in practice (Forrester and Keane (2009)). A common model example is one with quadratic effects and linear interactions for two input variables:

$$\hat{Y}(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 {x_1}^2 + \beta_4 {x_2}^2 + \beta_5 x_1 x_2 \tag{2.1}$$

Despite modelling non-linear effects in the design variables, the advantage of these models is that they linear in regards to the model parameters, β_i:

$$\hat{Y} = \beta\Phi \tag{2.2}$$

Where $\hat{Y}$ is the vector of observed response, $\Phi(.)$ is the Vandermonde matrix, and β the vector of model parameters. These are estimated by Ordinary Least Squares (OLS) which has an analytical solution, for which Maximum Likelihood Estimate (MLE) is given by:

$$\beta = (\Phi^T \Phi)^{-1} \Phi^T Y \tag{2.3}$$

Choosing the polynomial degree is not a simple task. While increasing the model order, and thus, the number of model parameters, usually results in better approximation, there is a risk of overfitting, and poor generalization. These models are usually unsuited for highly non-linear or high dimensional design spaces and may present limited use for more complex engineering problems, unless restricted to local regions (Forrester and Keane (2009)). For more information on polynomial models, the reader is referred to the book by Box and Draper (1987).

2.4.2 Gaussian process regression (Kriging)

Gaussian Process (GP) regression (Rasmussen and Williams (2006)), or Kriging (Sacks et al. (1989), Forrester et al. (2008), Bhosekar and Ierapetritou (2018)), is a non-parametric probabilistic model that approximates the function as a polynomial trend with residuals modelled by a zero-mean GP:

$$\hat{y}(x) = h(x)^T \beta + f(x) \tag{2.4}$$

Where $h(x)$ is a vector of linear independent basis functions which define the underlying trend, β the coefficients of these functions. The second term is a GP, which is formally defined as a collection of random variables, with a joint Gaussian distribution among a finite number of them (Rasmussen and Williams (2006)). It is completely specified by its mean, which in this case is considered zero, and covariance function, $k(.,.)$:

$$f(x) \sim \mathcal{GP}(0, k(x, x')) \tag{2.5}$$

In the literature three variants of Kriging can be distinguished by differences in the trend (Bhosekar and Ierapetritou (2018)). Simple Kriging assumes the trend to be a known constant, while Ordinary Kriging (OK) assumes this trend is constant but unknown. Universal Kriging approximates the trend as low order polynomial regression. For simplicity sake, the mathematical derivations are presented for the case of a zero mean constant trend and, considering deterministic simulations, noise free observations. The covariance function, also referred as kernel or correlation function, specifies the covariance between two random variables x and x':

$$k(x, x') = {\sigma_f}^2 \exp\left(-\frac{\|x - x'\|}{l}\right) \text{ (Exponential)} \tag{2.6}$$

$$k(x, x') = {\sigma_f}^2 \exp\left(-\frac{\|x - x'\|^2}{2l^2}\right) \text{ (Squared Exponential)} \tag{2.7}$$

$$k(x, x') = {\sigma_f}^2 \left(1 + \frac{\sqrt{3}\|x - x'\|}{l}\right) exp\left(-\frac{\sqrt{3}\|x - x'\|}{l}\right) \text{ (Matérn } 3/2) \tag{2.8}$$

$$k(x, x') = {\sigma_f}^2 \left(1 + \frac{\sqrt{5}\|x - x'\|}{l} + \frac{5\|x - x'\|^2}{3l^2}\right) exp\left(-\frac{\sqrt{5}\|x - x'\|}{l}\right) \text{ (Matérn } 5/2) \tag{2.9}$$

Where $\|.\|$ is the Euclidean norm, σ_f the input signal standard deviation and l the length-scale, which can be thought as the correlation strength between input variables. These covariance functions are stationary because their value is only a function of $\|x - x'\|$. It is also possible to consider a different l for each input variable, thereby implementing Automatic Relevance Determination (ARD) (Rasmussen and Williams (2006)). Considering a matrix of $D \times n$ of D variables with n training points, y and y_* vectors of training and testing outputs, respectively, their joint distribution is:

$$\begin{bmatrix} y \\ y_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{2.10}$$

The conditional probability, $P(y_* \mid X_*, X, y)$, indicates how likely the response y_* is, given the training data X, y and the observed data X_*:

$$P(y_* \mid X_*, X, y) \sim \mathcal{N}(K(X_*, X)K(X, X)^{-1}y,\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)^T) \quad (2.11)$$

The best estimate is given by the mean of this predictive distribution, $\hat{\mu}$ and the uncertainty of this estimate is given by its predicted variance $\hat{\sigma}^2$:

$$\hat{y} = \hat{\mu}(x) = K(X, X_*)K(X, X)^{-1}y \quad (2.12)$$

$$\hat{\sigma}^2 = K(X_*, X_*) - K(X, X_*)K(X, X)^{-1}K(X, X_*)^T \quad (2.13)$$

Where $K(X_*, X)$ denotes the $n \times n_*$ matrix of covariances evaluated at all pairs of training and test points, and similarly for $K(X, X)$, $K(X_*, X)$ and $K(X_*, X_*)$. The model parameters are estimated from training data, most commonly using Maximum Likelihood Estimate (MLE) or Cross Validation (CV). For convenience, the log marginal likelihood in MLE is used:

$$logp(y \mid X, \theta) = -\frac{1}{2}y^T K(X, X \mid \theta)^{-1}y - \frac{1}{2}log \mid K(X, X \mid \theta) \mid -\frac{n}{2}log2\pi \quad (2.14)$$

Where θ is the vector of model parameters. For a large amount of training data, the computation complexity of Equation (2.14) is high, due to the inversion of the covariance matrix. Equation (2.14) may present local optima, although this is generally not a significant problem (Rasmussen and Williams (2006)). For further details on GP regression, the reader is referenced to Rasmussen and Williams (2006). Other types of kriging models are also available in the literature, including co-kriging, which uses information of cheaper and more plentiful data from a lower fidelity source, and gradient-enhanced kriging, that incorporates gradient information from the black-box function if it is cheaply available. More details can be found in Forrester and Keane (2009) and Forrester et al. (2008).

2.4.3 Artificial Neural Networks

The development of Artificial Neural Networks (ANN) was motivated by the workings of the biological brain, which is capable of highly non linear processing, fast pattern recognition and high parallel computing (Haykin (2009)). ANN can be used for a multitude of different tasks but in chemical process engineering are mainly used for non linear regression and classification problems. These models present several advantages such as the ability to represent highly non linear processes with accuracy, model flexibility and robustness to measurement noise. Their importance and widespread acceptance in the chemical engineering field is indisputable, with already 200 plus references about their use cited by Himmelblau (2008), more than a decade ago. McCulloch and Pitts (1943) first introduced the single neuron as a computing machine and based on his work, Rosenblatt (1957) developed a single layer with a single

neuron for classification problems. ANN can be mainly divided into three different classes of networks: single-layer feedforward, with no hidden neuron layers, multilayer feedforward, with one or more hidden neuron layers, and recurrent (Haykin (2009)). In this work, only feed-forward networks with multiple layers are considered, for which a graphical representation is presented in Figure 2.6:

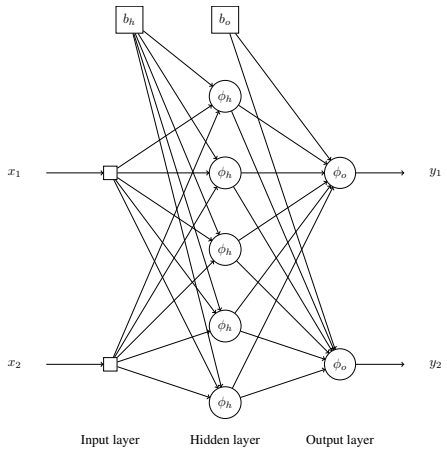


Figure 2.6: Neural network with one hidden layer with 5 neurons, 2 inputs and 2 outputs

Contrary to the previous model types, the flexibility of ANN enables the approximation of Multiple Input Multiple Output (MIMO) processes. For simplicity, mathematical expressions are provided considering a network with only one hidden layer:

$$\hat{y_i} = \phi_o(A\phi_h(Bx_i + b_h) + b_o) \tag{2.15}$$

Where $\hat{y}$ is the output vector corresponding to input vector x_i, ϕ_o and ϕ_h the activation functions of the output and input layers, respectively. A and B are the matrices of weights of the connections between layers, and b_h and b_o are the bias vectors of the hidden and output layer, respectively. Usually, the activation functions of the output layer are linear, while those of the hidden layer are sigmoid functions, graphically represented by an S-shape curve:

$$\phi(v_k) = \frac{1}{1 + e^{-av_k}} \qquad \text{(Logistic)} \tag{2.16}$$

$$\phi(v_k) = tanh(v_k) \qquad \text{(Hyperbolic Tangent)} \tag{2.17}$$

Where a is a slope parameter and v_k is the input to the $k - th$ neuron:

$$v_k = \sum_{i}^{m} w_{ik}x_i + b_k \tag{2.18}$$

w is the connection weight of the input i to the neuron k, x_i is a vector of the input variable

i values and m is the total number of input variables. These are the most popular activation functions, but others can also be used, as long as they are differentiable (Haykin (2009)). The first step in designing an ANN is choosing its architecture, that is, the number of hidden layers and neurons in each layer, the type of activation function and the connectivity between layers. This in an inherent downside to the flexibility of ANN, as the number of possible architectures is very high. It was shown that a network with a single hidden layer can approximate any continuous function, given enough individual neurons. However, it may require long training times and lead to poor generalization ability (Haykin (2009)). On the other hand, while increasing the number of layers can increase the network predictive ability, it requires larger training data sets and fitting times, due to the larger amount of parameters. There are no general rules for selecting the number of neurons and layers, so this choice is usually problem dependant and done by trial and error until model accuracy improves (Himmelblau (2008)). An alternative approach is to treat these hyper-parameters as decision variables in an optimization problem, which can be solved through sampling procedures based on extensive evaluating a regular grid, i.e. grid search, MC sampling or optimization algorithms, even including ones based on surrogate models (Bikmukhametov and Jäschke (2020)). After the network architecture is chosen, its parameters, weights and biases, are estimated by an error backpropagation algorithm which relies on optimization, usually derivative-based, to determine the weights, w, and biases, b, that minimize a loss function J, such as the Mean Squared Error (MSE):

$$\min_{w,b} J = \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n} \tag{2.19}$$

This objective function is usually highly non-convex (Himmelblau (2008)), which makes the optimization problem difficult and computationally expensive to solve, especially with bigger networks. For more information regarding ANN cf. Haykin (2009).

2.4.4 Radial Basis Function Network

A Radial Basis Function (RBF) model approximates a function as a linear combination of basis functions with weight coefficients:

$$\hat{y}(x) = \sum_{i=1}^{n} \lambda_i \phi(\|x - x_i\|) \tag{2.20}$$

Where n is the number of sampling points, x is a n-dimensional vector of design variables, x_i is a n-dimensional vector of design variables at i-th sampling point, called the centre of the basis function, and λ is the weight coefficient for the i-th basis function. This model can be interpreted as a special type of multilayer feedforward neural network with no input weights and a single hidden neuron layer where the centres are the neurons. Each training data point is a centre, so the layer has size equal to the number of training points. The activation function is

a radial basis function, for which there are several options:

$$\phi(\|x - x_i\|) = \|x - x_i\| \quad \text{(Linear)} \tag{2.21}$$

$$\phi(\|x - x_i\|) = \|x - x_i\|^3 \quad \text{(Cubic)} \tag{2.22}$$

$$\phi(\|x - x_i\|) = \|x - x_i\|^2 log(\|x - x_i\|) \quad \text{(Thin-plate spline)} \tag{2.23}$$

$$\phi(\|x - x_i\|) = \exp^{-\gamma\|x - x_i\|^2} \quad \text{(Gaussian)} \tag{2.24}$$

$$\phi(\|x - x_i\|) = \sqrt{\|x - x_i\|^2 + \gamma^2} \quad \text{(Multi-quadratic)} \tag{2.25}$$

In the equations above, γ is a positive used defined constant. This is a hyper-parameter that improves generalization properties at the expense of model fitting complexity (Forrester and Keane (2009)). If each training data point is chosen as a centre, the coefficient λ can be estimated considering the interpolation condition:

$$y = \hat{y} \leftrightarrow \lambda = \Phi^{-1} y \tag{2.26}$$

Where y is a vector of n observations and Φ an $n \times n$ interpolation matrix $\Phi_{i,j} = \phi(\|x_i - x_j\|), i, j = 1, 2, ..., n$. Because of this RBF models are exact interpolators. The RBF function can also be augmented to include a polynomial term (Gutmann (2001),Fang and Horstemeyer (2006),Bhosekar and Ierapetritou (2018)):

$$\hat{y}(x) = \sum_{i=1}^{n} \lambda_i \phi(\|x - x_i\|) + \sum_{j=1}^{m} \beta_j p_j(x) \tag{2.27}$$

Where p is a polynomial function, β the vector of polynomial coefficients and m the total number of terms in the polynomial. Since the equation is undetermined, orthogonality is imposed between the vectors λ and β. By combining this condition with the equation above we have the following system of equations:

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \beta \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix} \tag{2.28}$$

Where $P_{i,j} = p_j(x_i), i = 1, ..., n, j = 1, ..., m$ is a matrix with the values of the polynomial terms. The weights, λ, and polynomial coefficients, β, can be determined by solving the linear equation system above.

2.4.5 Support Vector Regression

Support Vector Regression (SVR) arises from theory of Support Vector Machine (SVM), mainly used for classification problems (Haykin (2009)). The SVR prediction takes a form

similarly to that of kriging or RBF:

$$\hat{y(x)} = \mu + \sum_{i=1}^{n} w_i \psi(x, x_i) \tag{2.29}$$

Where $\hat{y}$ is the predicted value, μ is a bias, w is the weight coefficient, $\psi(.)$, which can take several forms (Haykin (2009), Forrester and Keane (2009)):

$$\psi(x_i, x_j) = {x_i}^T x_j \qquad \text{(Linear)} \tag{2.30}$$

$$\psi(x_j, x_j) = ({x_i}^T x_j + 1)^d \qquad \text{(Polynomial)} \tag{2.31}$$

$$\psi(x_i, x_j) = exp\left(\frac{-\|x_i - x_j\|^2}{\gamma}\right) \qquad \text{(Gaussian)} \tag{2.32}$$

Where d is a polynomial degree and γ is a user defined constant. Despite the similarities with other models, the SVR parameters, w, and μ, are estimated by solving a quadratic optimization problem:

$$\min_{(\alpha^+ - \alpha^-)} \frac{1}{2} \sum_{i,j=1}^{n} (\alpha^{+(i)} - \alpha^{-(i)})(\alpha^{+(j)} - \alpha^{-(j)})\Psi(x_i, x_j) + \epsilon \sum_{i=1}^{n} (\alpha^{+(i)} - \alpha^{-(i)}) - \sum_{i=1}^{n} y_i(\alpha^{+(i)} - \alpha^{-(i)}) \tag{2.33a}$$

s. t.

$$\sum_{i=1}^{n} (\alpha^{+(i)} - \alpha^{-(i)}) = 0 \tag{2.33b}$$

$$0 \leq \alpha^+, \alpha^- \leq \frac{C}{n} \tag{2.33c}$$

The use of a kernel enables that mapping its inner product into a feature space, so the objective function remains linear and the optimization problem becomes convex. ϵ is a constant that indicate the approximation error tolerance, which means that samples with an error smaller than this value will have no loss associated with them. C is a constant that regulates the trade-off between model complexity and the degree to which errors larger than ϵ are tolerated. The decision variable is the difference between the values of α, which are Lagrange multipliers. If this difference is different than zero for a given sample, that point is considered a support vector and used to define the margins between $+\epsilon$ and $-\epsilon$. By substituting the initially defined w for this value, the SVR prediction is:

$$\hat{y(x)} = \mu + \sum_{i=1}^{n} (\alpha^{+(i)} - \alpha^{-(i)})\psi(x, x_i) \tag{2.34}$$

μ can be calculated by one of two equations:

$$\alpha^{+(i)} = 0 \text{ and } \mu = \sum_{i=1}^{n}(\alpha^{+(i)} - \alpha^{-(i)})\psi(x_i, x_j) + \epsilon, \text{ if } 0 \leq \alpha^{-(i)} \leq \frac{C}{n} \tag{2.35}$$

$$\alpha^{-(i)} = 0 \text{ and } \mu = \sum_{i=1}^{n}(\alpha^{+(i)} - \alpha^{-(i)})\psi(x_i, x_j) - \epsilon, \text{ if } 0 \leq \alpha^{+(i)} \leq \frac{C}{n} \tag{2.36}$$

The main hyper-parameters of SVR are the kernel, and constants ϵ and C. The value of ϵ can be determined considering the standard deviation of the training data noise, when it is known (Forrester and Keane (2009)). However, considering a deterministic simulation scenario, in the present work this parameter is considered to be zero. While there is usually an optimum value for C, the exact choice is not overly critical, so it is sufficient to test several values of different magnitudes and see their impact on the prediction error (Forrester and Keane (2009)). The selection of these hyper-parameters can also be done in the same way as mentioned for ANN. For more details regarding mathematical derivations and recommendations c.f. Smola and Schölkopf (2004) and Forrester and Keane (2009), respectively.

2.5 Surrogate model validation

After training the model, a critical step is deciding if it has sufficient quality to accomplish the objectives for which it was created. The procedure to assess the model quality is referred to as validation (Bhosekar and Ierapetritou (2018)). Beyond testing the accuracy of the surrogate model, validation techniques can also be used to tune its hyper-parameters (e.g. number of layers in ANN, kriging or SVR kernel). The validation procedure requires an additional set of data, referred to as the test set. When the possibility of acquiring more data is not available, resampling strategies such as cross validation or bootstrapping can also be used (Bhosekar and Ierapetritou (2018)). The error between the output variable values in this set and the predicted output variables is used as a validation metric. In this work, model validation is accomplished by using an additional test data consisting of n_* samples and calculating the Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n_*}(y_{*i} - \hat{y}_i)^2}{n_*}} \tag{2.37}$$

For additional surrogate model validation metrics and resampling procedures, cf. Bhosekar and Ierapetritou (2018).

Chapter 3

Global optimization of black-box problems using surrogate models

In the first section of this chapter, the concept of Derivative Free Optimization (DFO) is introduced along with references to some commonly used methods. The following section describes DFO methods that rely on both local and global surrogate models, describing in detail two methods used in this work. A brief introduction to multi-objective optimization is made in the third section. The case studies considered in this work only include non-constrained optimization problems with continuous decision variables. Therefore, for the sake of brevity, topics such as how to incorporate inequality constraints and integer design variables in Surrogate-Based Optimization (SBO) are not discussed. The reader is referred to Forrester and Keane (2009), Gelbart et al. (2014), Boukouvala et al. (2016) and Greenhill et al. (2020).

3.1 Derivative free optimization of black box functions

A black-box, or when partial information is available, grey-box, optimization problem can be defined as (Boukouvala et al. (2016)):

$$\min_{x} \; f(x) \tag{3.1a}$$

s.t.

$$h_k(x) = 0 \tag{3.1b}$$

$$g_k(x) \leq 0 \tag{3.1c}$$

$$h_{uk}(x) = 0 \tag{3.1d}$$

$$g_{uk}(x) \leq 0 \tag{3.1e}$$

$$x_{lb} \leq x \leq x_{ub}, \; x \in \mathbb{R}^D \tag{3.1f}$$

Where $f(x)$ is the objective function to minimize, x is the vector of D-dimensional decision variables, with upper and lower bounds x_{ub} and x_{lb}, respectively, h and g are the equality and inequality constraints. The subscripts k and uk indicates if their algebraic expressions are known or unknown, respectively. No assumptions can be made in regards to the form of $f(x)$ including convexity properties, and explicit closed form derivative expressions with regards to the design variables are unavailable. When the underlying system is also considered computationally expensive, estimating them numerically is prohibitive. Discontinuities may also exist, causing derivative information to be unreliable (Boukouvala et al. (2016)). Because of this, using methods based on finite differences are generally of little or no use in these situations (Rios and Sahinidis (2012)). A solution for this type of problems is the use of Derivative Free Optimization (DFO) methods. One important note must be made, contrary to what the previous designation may imply, algorithms used to solve this class of problems may use derivative information of another function other than the original.

DFO methods usually rely on direct search, through objective function evaluations, to find the optimum and can be classified into several categories: the ones that use either local or global search, deterministic or stochastic methods and model-free or model-based methods (Rios and Sahinidis (2012)).

Global deterministic methods include the DIRECT method (Jones et al. (1993)), while local methods include the Nelder and Mead simplex (Nelder and Mead (1965)), and commonly used global stochastic include evolutionary algorithms such as Genetic Algorithm (GA) (Holland (1975)) and Particle Swarm Optimization (PSO)(Kennedy and Eberhart (1995)). An in depth review of the state of the art on DFO is out of the scope of this thesis, so the reader is referred to Rios and Sahinidis (2012) or (Boukouvala et al. (2016)) for further information.

The main disadvantage of popular DFO methods such as GA or PSO is that they typically require a significant number of objective function evaluations to converge (Boukouvala et al. (2016)). This can make their use impractical for expensive simulations. In these cases, the use of model-based methods to solve expensive DFO problems is advantageous, since these methods usually require less objective function evaluations to find the global optimum.

The problem with optimization of black-box functions is that because no derivative information is available, theoretical optimality properties such as the Karush-Khun-Tucker (KKT) conditions cannot be applied directly to determine if a point is a local optimum. In addition, when using global DFO methods convergence to the global optimum is also not guaranteed. However, the motivation is usually to find a "good enough", or close to optimal, solution given enough computational resources (Yang et al. (2018)).

3.2 Derivative Free Optimization using Surrogate Models

Model-based methods or Surrogate-Based Optimization (SBO) as the name implies, rely on a surrogate model of the original objective function that is used to guide the search for the global optimum in a pre determined bounded domain or design space. There are two main approaches to using model-based methods in optimization: using local or global models (Rios and Sahinidis (2012), Boukouvala et al. (2016), Bhosekar and Ierapetritou (2018)).

In local, or trust-region, model-based methods, a surrogate model is built in the neighbourhood, referred to as a trust region, of a sample in the design space, where it is expected to be accurate. The radius of this trust region is iteratively refined by contraction or expansion, according to a defined metric, taking into account the current iteration optimum value. When this radius decreases below a given threshold, the algorithm terminates. Because these methods are general, several model types can be used, such as linear interpolations (Powell (1994)), quadratic interpolations (Powell (2009)) or even RBF (Regis and Wild (2017)). A kriging based Efficient Global Optimization (EGO) method was also used in a trust-region framework (Regis (2016)). It has been shown that these local-search methods can guarantee first-order Karush-Khun-Tucker (KKT) conditions under certain assumptions, as the gradients of the surrogate model approach those of the black-box function and the trust region radius tends to zero (cf. Biegler et al. (2014), Eason and Biegler (2016) and references therein). However, despite their attractiveness, trust-region methods only guarantee local optimality at best, as they do not incorporate exploration (Jones (2001), Forrester and Keane (2009)). They can still be used for global optimization if a multi-start approach is employed (Regis and Shoemaker (2007)), but an alternative is to use model-based methods that aim to approximate the entire design space.

Global model-based methods rely on a surrogate model that approximates the entire design space or multiple parts of the design space. An adaptive sampling procedure is used to refine the surrogate model and guide the search for the optimum. As demonstrated by Jones (2001), using pure exploitation by simply minimizing the surrogate model and iteratively using the predicted optimum as an adaptive sample does not necessarily lead to convergence to even a local optimum. A better alternative is to, as described in Chapter 2.2.2, use a balance of local exploitation, to improve the current optimum and global exploration, to escape local optima and to not disregard regions where the global optimum might be located. The most popular global model-based methods use either a kriging or a RBF surrogate model, due to their interpolating ability. The former methods usually consider the optimization of an acquisition function that balances exploration and exploitation (Jones et al. (1998), Jones (2001), Brochu et al. (2010), Frazier (2018)).Those using RBF usually rely on the minimization of a bumpiness function (Gutmann (2001)) or a merit function and MC sampling (Regis and Shoemaker (2007), Wang and Shoemaker (2014)).

Also worth mentioning is GLIS, recently developed by Bemporad (2020), uses a combi-

nation of RBF and Inverse Distance Weighting (IDW). It showed comparable performance to a commercial (MATLAB®) implementation of BO for a variety of test functions and a hyper-parameter tuning problem, while being computationally lighter.

For simplicity sake, only the methods used in this work, BO and MSRS, will be discussed in further detail. These methods were chosen due to popularity and the availability in user friendly commercial implementation in MATLAB® toolboxes. Specifically, BO is implemented in *bayesopt* in the Statistics and Machine Learning Toolbox^{TM} and MSRS in *surrogateopt* in the Global Optimization Toolbox^{TM}.

3.2.1 Bayesian Optimization using Gaussian process regression

The term Bayesian Optimization (BO) (Brochu et al. (2010)) is typically used to describe methods that employ a kriging surrogate model on the objective function and an acquisition function with the purpose of finding the global optimum in a bounded domain. The estimated prediction variance is used to formulate a probabilistic framework in the adaptive sampling procedure. It was first popularized through the Efficient Global Optimization (EGO) algorithm, developed by Jones et al. (1998). A short general algorithm is presented in 1.

Algorithm 1: Pseudo Algorithm for Bayesian Optimization

Result: Value x^*that minimizes f(x)
1. Generate initial samples X
2. Evaluate the objective function f(x) for each sample in X
3. Fit a Kriging model on f(x)
while $N < N_{max}$ **do**
  4.1. Find the value x^* that optimizes the acquisition function
  4.2. Evaluate the objective function on x^* and obtain $f(x^*)$
  4.3. Augment the training dataset with x^* and $f(x^*)$ and refit the kriging model
end
5. Return best found value x^*

A initial set of samples is generated using a design such as the ones mentioned in Chapter 2. The objective function is evaluated on those samples and a kriging model is then fitted to approximate the input-output relationship. The kriging trend and kernel are not fixed and can be chosen based on user preference or underlying system information. The algorithm then begins an iterative search for the next sample point that relies on the optimization of an acquisition function, or infill criteria (Rojas-Gonzalez and Van Nieuwenhuyse (2019)). This step is the core of BO. The most common acquisition functions are the Lower Confidence Bound (LCB), Probability Of Improvement (POI) and Expected Improvement (EI). Other entropy-based infill criteria (Hennig and Schuler (2012), Hernández-Lobato et al. (2014)), can also be used, but these are not considered in this work. Since the acquisition function formulation depends on whether the goal is maximization or minimization, is important to clarify that this work consid-

ers a minimization problem.

Cox and John (1997) introduced a method for selecting evaluation points based on the minimization of a statistical Lower Confidence Bound (LCB) of the prediction:

$$LCB(x) = \hat{\mu}(x) - k\hat{\sigma}(x) \tag{3.2}$$

Where $\hat{\mu}$ and $\hat{\sigma}$ are the predicted mean and standard deviation, respectively. k is a user positive constant that controls the exploration-exploitation trade-off. A larger value favours exploration while a lower one favours exploitation. Choosing an adequate value for k presents a disadvantage in using this acquisition function (Forrester et al. (2008)). In *bayesopt*, this value is set to 2 by default.

The following acquisition functions consider a measure of improvement, I, over the current best value, f_{min}, to select the next sample, x:

$$I(x) = f_{min} - f(x) \tag{3.3}$$

Given I, the Probability Of Improvement (POI) criterion is used to maximize the probability that the next sample x results in a improvement below a target, ζ<f_{min}:

$$P[I(x) \leq \zeta] = P(f(x) < I(x)) = \begin{cases} \Phi\left(\frac{f_{min}-\hat{\mu}(x)-\zeta}{\hat{\sigma}(x)}\right), & \hat{\sigma}(x) \neq 0 \\ 0, & \hat{\sigma}(x) = 0 \end{cases} \tag{3.4}$$

Where $\Phi(.)$ is the normal Cumulative Distribution Function (CDF). While the use of this criterion can result in global convergence (Forrester et al. (2008)), the main disadvantage is that no quantitative measure of the improvement is considered, only the probability that x provides one, which is highly sensitive to the choice of the target (Jones (2001)). A low ζ can result in highly local search being done in the vicinity of a local optimum before exploring other regions while a high ζ can result in excessive global search before improving promising solutions. To avoid these issues, it is suggested to cycle through several values of ζ (Jones (2001)), however, a more practical approach may be to use the an infill criterion that directly considers a magnitude of the improvement.

Instead of maximizing the probability of an improvement existing, POI, Jones et al. (1998) suggests maximizing the expectation of I, which can be evaluated analytically by:

$$E[I(x)] = \begin{cases} (f_{min} - \hat{\mu}(x))\Phi\left(\frac{f_{min}-\hat{\mu}(x)}{\hat{\sigma}(x)}\right) + \hat{\sigma}(x)\phi\left(\frac{f_{min}-\hat{\mu}(x)}{\hat{\sigma}(x)}\right), & \hat{\sigma}(x) \neq 0 \\ 0, & \hat{\sigma}(x) = 0 \end{cases} \tag{3.5}$$

Where $\phi(.)$ is the normal Probability Density Function (PDF). The advantage of EI is that, contrary to the other acquisition functions, no parameter needs to be tuned and because a

high $\hat{\mu}$ results in exploitation and a high $\hat{\sigma}$ favours exploration, the trade off between the two is automatically established. However, difficulties arise when using the EI criteria, because Equation (3.5) can present local optima that difficult the adaptive sample search (Bhosekar and Ierapetritou (2018)). Due to this, the acquisition function optimization may itself become the limiting step in some cases.

The optimization algorithm used for the acquisition function maximization impacts both optimum search results and required time. Since the kriging model is assumed to be evaluated cheaply, an algorithm that requires many evaluations can be used. Brochu et al. (2010) recommends DIRECT, while algorithms referenced in Rojas-Gonzalez and Van Nieuwenhuyse (2019) use GA. MATLAB®'s *bayesopt* uses the Nelder and Mead Simplex through *fminsearch*.

The objective function is evaluated at the optimum value of the acquisition function, this sample is added to the initial set and the model is refitted. A new iteration begins, until a predetermined stopping criterion is met. This can be related to time spent or maximum number of objective function evaluations. Jones et al. (1998) also suggested stopping the search when the difference of the value of EI is less than a given tolerance than the current best objective function value. Because of the possibility of over exploitation in a local minimum, defining convergence based on the acquisition does not guarantee that the global optimum is found. This means that defining a convergence stopping criterion is subjective (Forrester and Keane (2009)).

Despite the advantages of BO, this approach suffers from practical limitations in high dimensional problems, with more than 20 decision variables (Greenhill et al. (2020)). This is shown in Wang and Shoemaker (2014), where EGO resulted in worse performance against other algorithms for test problems with more than 30 design variables.

3.2.2 Stochastic Response Surface using Radial Basis Functions

Metric Stochastic Response Surface (MSRS) was developed by Regis and Shoemaker (2007) and similarly to BO, refines a surrogate model using an auxiliary function to determine the selection of adaptive points. The two main differences is that this method uses a non probabilistic RBF model, and instead of solving an auxiliary optimization problem to find the adaptive sample, MSRS uses MC sampling to generate a large number of candidate points, evaluating them on a merit function. The one with the best value is then selected as the next value to sample the objective function. The assumption is that RBF model is fast to evaluate on a high number of candidate points and can therefore result in time efficiency when compared to an optimization based search approach such as BO (Regis and Shoemaker (2007)). The MATLAB® implementation, *surrogateopt*, used in this work uses the Multistart Local MSRS algorithm, which showed superior results in problems up to 14 dimensions compared to other algorithms (Regis and Shoemaker (2007)). A description of the algorithm is made in 2.

Algorithm 2: Pseudo Algorithm for Local MSRS

Result: Value x^* that minimizes $f(\mathrm{x})$
1. Generate initial samples X
2. Evaluate the objective function $f(\mathrm{x})$ on each sample in X
3. Fit a RBF model on $f(\mathrm{x})$
while $N < N_{max}$ **do**
  4.1. Generate thousands of pseudo-random normally distributed candidate points around best found value of x^*, or incumbent
  4.2. Estimate output data values using the RBF approximation of the objective function
  4.3. Calculate merit function value for each candidate point
  4.4. Evaluate original objetive function on the candidate with highest merit function value $f(x^*)$
  4.5. Augment the training dataset with x^* and $f(x^*)$ and refit the RBF model
  if *All x are whitin minimum distance* **then**
    Go back to step 1
  end
end
5. Return best found value x^*

Similarly to BO, a set of initial samples is generated using DoE, evaluated on the objective function and a surrogate model is fitted. *surrogateopt* uses a cubic RBF model with a linear polynomial term. The algorithm begins a loop where, at each iteration, a set of hundreds or thousands of candidate points is generated. Contrary to the Global MSRS algorithm, which randomly generates these candidates uniformly across the whole design space, the Local MSRS generates these points by adding normally distributed perturbations around the best current solution, or incumbent. Those candidate points are evaluated on a merit function that considers both a weighted sum of two scaled criteria:

$$f_{merit}(x) = w\frac{\hat{f}(x) - \hat{f}_{min}}{\hat{f}_{max} - \hat{f}_{min}} + (1 - w)\frac{d_{max} - d(x)}{d_{max} - d_{min}} \tag{3.6}$$

In the first term of the above equation, $\hat{f}(x)$ is the estimated objective function value of the candidate point and $\hat{f}_{max}$ and $\hat{f}_{min}$ are the estimated maximum and minimum objective function values of all the candidate points, respectively. The second term is a distance criterion where $d(x)$ is the minimum distance between the candidate point and previously evaluated samples, d_{max} is the maximum minimum distance and d_{min} and the minimum value of $d(x)$. w is a weight that varies between 0 an 1 and determines the trade-off between exploration and exploitation. A higher value gives more importance to exploitation while a lower value assigns favours exploration. The point with the minimum merit function value amongst all candidates is selected as the next sample to evaluate on the objective function. The dataset is then augmented with both values, a RBF model is refitted to that dataset and a new iteration begins.

The Local MSRS can be thought as a local search algorithm, exploring the neighbourhood

of the current best solution, which can result in convergence to a local minimum. To avoid this situation, global search can be achieved by restart the algorithm when it appears to have locally converged. The algorithm restarts by generating a new set of initial samples completely, as to avoid bias in the optimization trajectory (Regis and Shoemaker (2007)). *surrogateopt* generates this new set of samples using a QRLD sequence. The algorithms ends when a maximum number of objective function evaluations is achieved.

MSRS was extended to handle higher dimensional (above 30 design variables) problems with the algorithms DYCORS (Regis and Shoemaker (2013)) and SO-SA (Wang and Shoemaker (2014)). In the latter reference, a comparison between different algorithms for high dimensional test problems was made.

3.3 Multi-objective surrogate based-optimization

Multi-objective optimization refers to problems where two or more objectives are simultaneously present:

$$\min_x [f_1(x), f_2(x), ..., f_m(x)] \tag{3.7a}$$

s.t.

$$x_{lb} \leq x \leq x_{ub} \tag{3.7b}$$

Where m is the total number of objectives. Generally, these can be conflicting, i.e., a solution may improve one objective while worsening another. In these problems there is no single global solution, but instead a set of optimal solutions to choose from. The concept of Pareto optimality is used to define these optimal solutions (Marler and Arora (2004)). A point is Pareto optimal if there is no other point that improves all of the objective functions simultaneously and the set of Pareto optimal points forms the Pareto set. The solution is chosen from this set by articulating the preference, or relative importance, of the different objectives, which can be articulated *a priori*, *a posteriori* or even not at all (Marler and Arora (2004)). In the first category of methods, the user specifies a preference function beforehand that aggregates the multiple objectives into a single one, which is solved by traditional single objective optimization methods. In *a posteriori* methods, the Pareto set is approximated and the solution is chosen from this representation based on the user preference. For additional information, the reader is referred to Marler and Arora (2004) for review of multi-objective optimization methods and Rangaiah et al. (2020) for applications and recommendations in chemical engineering.

When the multiple objectives are converted into a single one, standard optimization algorithms can be used. However, the approximation of the Pareto front is commonly done using evolutionary algorithms, such as GA or PSO(Marler and Arora (2004), Rangaiah et al. (2020)). Since these require a high number of function evaluations, their use for multi-objective optimization problems that require expensive simulations can lead to a large computational burden, motivating the use of surrogate models. These can be used as a standalone optimization approach or in conjunction with evolutionary algorithms, see Díaz-Manríquez et al. (2016) or Chugh et al. (2017) for a review.

When using BO for multi-objective optimization problems, two approaches can be taken (Rojas-Gonzalez and Van Nieuwenhuyse (2019)): either considering the previously mentioned single objective infill criteria and aggregating the multiple objectives into a single one, or considering specific criteria for multiple objectives. An extensive discussion on BO algorithms for multi-objective optimization of is made by Rojas-Gonzalez and Van Nieuwenhuyse (2019). When dealing with multi-objective optimization, contrary to BO, the MSRS method can only be used by scalarization of the multiple objectives into a single objective function.

In this work, the multi-objective problem is transformed into a single objective by defining the preference *a priori*. While several methods exists, the most common, straightforward and simple is the weighted sum:

$$\min_{x} \sum_{j=1}^{m} w_j f_j(x) \tag{3.8}$$

w represents the objective weight vector of length m, and the choice of these values reflects the preference of the decision maker towards each objective. A larger weight reflects higher importance. Despite the simplicity of this method, it possesses several drawbacks (Marler and Arora (2004)). Selecting the weights to reflect desired preference is non-trivial and one may have to re-solve the optimization problem with new weights to find an acceptable solution. Additionally, differences in objective scales may result in the weights being used to mask a scaling problem, instead of assigning preferences. This is especially problematic when considering ranking methods, which order the weight values according to preference (Marler and Arora (2010)). Another drawback is that if the Pareto set is not convex, there are no guarantees that the weighted sum method finds all Pareto optimal points. Finally, even if the weights are consistently varied, an uneven distribution of the Pareto front may arise. Nonetheless, this method is easy to use and applicable for situations where the preference does not need to be precisely defined. For more information on the method, cf. Marler and Arora (2004), Marler and Arora (2010) and references therein.

It is important to state that while specific BO algorithms that use scalarization exist, in this work, this is done outside the *bayesopt* algorithm, to make performance comparison with *surrogateopt* fair.

Chapter 4

State of the art on surrogate modelling in process control

In this chapter, applications of surrogate models, with an emphasis on surrogate-based optimization, for process control will be reviewed in detail. Surrogate models have recently been used to reduce complexity in optimization frameworks for process control and related fields, such as integration of control with planing and scheduling (Dias and Ierapetritou (2020)) and process design (Rafiei and Ricardez-Sandoval (2020)). However, in this thesis the focus is restricted to controller design and tuning, so the aforementioned topics are out of the scope of this work. Also, special emphasis is given to PID and MPC tuning, and less to explicit-MPC.

4.1 Optimal controller tuning and design

4.1.1 Proportional Integral Derivate (PID) control

Proportional integral derivative (PID) controllers are linear, parametric controllers widely accepted in industrial process control applications and have been so for decades (O'Dwyer (2009), Yu (2006)). However, despite their long time application, surveys have shown that their performance in practice is still subpar, with only a fraction of the controllers displaying good or even acceptable control performance (cf. O'Dwyer (2009) and Yu (2006) and references therein). Lack of controller performance can be attributed to several factors, one of which is poor tuning. Because the majority of regulatory layer controllers is of the PID type and these also serve as the foundation of more advanced control strategies such as Model Predictive Control (MPC) (Figure 1.2), their optimal design is a relevant concern in the process industries. PID controllers can generally be tuned using either model-based or model-free methods (Seborg et al. (2017)).

Model-based methods rely on the use of a dynamic model of the process, generally a linear model, such as a low order transfer function, to estimate the controller parameters. These

parameters are typically determined by deriving an analytical expression for the controller considering the desired closed-loop response (Direct Synthesis (DS)), by considering an internal process model within the closed loop model (Internal Model Control (IMC)) or by analytical expressions that relate the controller parameters with those of the process model, known as controller tuning relations (O'Dwyer (2009)). Additionally, the model could be used to obtain the system frequency domain response, which can be used to tune the controller, but these techniques are not considered in this work.

Because a mathematical process model always presents at least some degree of inaccuracy, plant physical tests are always required. Model-free methods do not require a process model and estimate the controller parameters based on the system closed loop response subject to different experimental tests, some times requiring extensive manual tuning. Because the system's real time response is used to tune the controllers, these methods are also referred as on-line or field-tuning (Seborg et al. (2017)). In an effort to reduce the amount of manual tuning, systematic approaches were developed, including the Continuous Cycling (Ziegler and Nichols (1942)) and the Relay feedback auto-tuning (Åström and Hägglund (1984)). These methods use a controller to induce an sustained oscillatory response of the system, from which ultimate frequency parameters are estimated and used in heuristic tuning relations to determine the controller parameters. In the first method, the controller is placed on Proportional (P) mode and, after a small setpoint change, the controller gain is iteratively increased until a sustained oscillation occurs. In the second method, this oscillatory response is obtained by temporarily replacing the PID with an on-off controller, or relay. The latter method has been favoured in an industrial setting because of its simple implementation, ability to be easily automated and the requirement of a single experiment (Yu (2006)). In addition, the process is not pushed to a stability limit (Seborg et al. (2017)). The ability to provide automatic tuning is especially relevant when there are hundreds or thousands of control loops, since manually tune each controller would require a great deal of time and effort.

If a model of the process is not available, one suitable for the purpose of controller tuning can be derived though an open loop step test on the physical plant. In this test, the controller is changed to manual operation, a step change is made on the manipulated variable, and the process model parameters are estimated based on the response, or reaction curve (Ziegler and Nichols (1942), Seborg et al. (2017)). The main disadvantage of this procedure is that due to open loop operation, it is not applicable for open-loop unstable processes. Also, the process may deviate significantly from the desired nominal operating range.

It is relevant to note that there has been an effort to establish methodologies that enable the estimation of a low order linear model of the system based on closed-loop experimental tests. The motivation is to gain system knowledge with smaller induced deviations from nominal operation conditions and to avoid safety issues due to instability caused by the experiments, especially when the system is open-loop unstable (Liu et al. (2013)). This can generally be

done either through a closed-loop step test or the Relay feedback method.

In the first test, a step change is made in the controlled variable setpoint and knowing the response and the controller structure, a process model is then estimated. In the second method, the characteristics of the oscillatory response under relay control are used to estimate the model. Once an approximate model is available, model-based controller tuning relations can be used (Marchetti et al. (2001), Yu (2006), Berner et al. (2016)). For an extensive review of closed-loop system identification methods using step and relay tests, consult Liu et al. (2013).

Also worth mentioning is the article by Berner et al. (2018), were the performance of two commercially available and widely used PID autotuners are compared with recently developed methods. These rely on system identification trough a relay test on the physical process, followed by model-based tuning methods. It was shown that present day computing resources allow an opportunity for improving the current performance of industrial autotuners, which are largely based on ideas derived decades ago, when such resources were not available.

A dynamic process model can also be used in computer simulation to gain insight on the process dynamic behaviour, to test different control scenarios and structures or to compare between controller tuning strategies, including, but not limited to, the previous methods. If this model is highly complex or its mathematical structure is unknown such that a low order linear model is difficult to derive mathematically, the previously mentioned system identification techniques can also be used in simulation.

When an sufficiently accurate process model exists, computer simulation can also be used to tune controllers through a more systematic and automated approach that relies on optimization (Åström and Hägglund (2006)). This is the general idea behind controller tuning relations based on the minimization of an integral error criteria. The controller parameters are optimized for a set of different low order transfer function process models with varying parameter values. Tuning relations are then developed by regressing the optimum controller parameters as a function of the process model parameters (Visioli (2001), Åström and Hägglund (2006), O'Dwyer (2009)). Despite providing useful benchmarks, these tuning relations are generally not robust (Seborg et al. (2017)). Lack of robustness arises due to several reasons, including uncertainty in the model parameters or process non-linearity. While the former can be reduced by including quantitative robustness metrics as constraints in the optimization problem, a practice commonly disregarded (Åström and Hägglund (2006)), non-linearity can be taken into account by considering a non-linear process model in the simulated closed-loop response. This can include a data driven, a first principles model or a combination of the two. An optimization problem can be then formulated where the controller parameters are optimized to minimize an objective function considering a combination of suitable quantitative metrics. A simplified block diagram of this approach is presented in Figure 4.1.

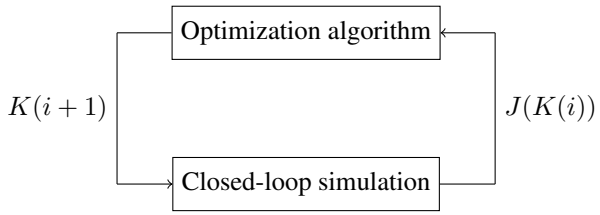


Figure 4.1: Flowchart of simulation optimization tuning (K - vector of controller parameters, J - performance function, i - current iteration)

The advantage of using simulation based optimization is that the controller parameters are optimally and automatically determined according to the desired criteria. The flexibility to consider multiple objectives also enables tuning other types of controllers such as cascade controllers or multiple interactive single loop controllers. This while enabling the use of a simulated response from a more rigorous process model, including those available in external commercial simulation platforms such as Aspen Plus®. This approach can therefore reduce trial and error while using higher fidelity simulation experiments, inevitably saving time and reducing engineering effort. There are, of course, disadvantages. Because multiple objectives are simultaneously considered, making this a multi-objective optimization problem, the objective function and the trade-off between different conflicting objectives is difficult do formulate. Also, because no explicit derivative information is available, DFO methods are recommended. Popular stochastic algorithms can be found in the literature, including GA (Visioli (2001)) and PSO (Latha et al. (2013)). However, these methods are usually inefficient in terms of objective function evaluations, which can be costly when the simulation experiment is time consuming. This is especially the case when the process non-linear model is high order and complex, described by a high number of differential equations. Recent years have shown an increase in the use of surrogate-based optimization, especially BO, to optimally tune and design controllers.

Yang et al. (2015) compared two multi-objective BO algorithms, EHVI-EGO and SMS-EGO, with an evolutionary algorithm to optimally tune a PID controller for two linear plant models. The optimization objective was to minimize both settling time and percentage of overshoot in a servo control problem. Both surrogate based algorithms, particularly EHVI-EGO, out-performed the evolutionary algorithm for a fixed maximum number of function evaluations, both dominating the average and best Pareto front. This means that surrogate based algorithms can converge faster to the "ideal" Pareto front with fewer evaluations.

In addition to single-loop PID, BO has also been applied to tune cascade controllers by Khosravi et al. (2020) for a axis drive model obtained using first principles and system identification. The controller structure consisted of a cascade sequence, with a PI and P controller in the inner and outer loops, respectively. The tuning procedure was sequential, first tuning the inner loop controller, then fixing its parameters and tuning the outer loop controller. The tuning problem was formulated as a multi-objective optimization scalarized by the weighted sum method. BO was compared with grid search, Z-N method, Relay autotuning and ITAE criterion.

Controllers tuned by BO displayed superior performance than traditional tuning methods and with almost identical gains as those obtained by grid search, which requires extensive function evaluations. It was also shown that the size of the initial sample had an impact on the number of iterations needed for BO to converge.

It is important to state that while the use of simulation optimization methods for control system tuning can reduce time and aid the decision-making process, it does not exclude tests on the physical system. At best, these methods can provide a good first estimate for fine tuning the controller parameters on site, if necessary. Also, the process model must be sufficiently accurate for the simulations to provide meaningful results.

Interestingly, surrogate-based optimization, especially BO, has also been used as an alternative to conventional physical experiments in on-line automatic controller tuning. In this model-free method, the relation between controller parameters and the objective function is approximated using surrogate models based on the direct measurements of the closed loop response of the physical process. Adaptive sampling is then used to suggest new controller parameters to be tested. Because the algorithm is efficient in regards to used data, the assumption is that it can optimize the controller performance given a limited number of costly physical experiments. However, due to the exploratory nature of the algorithms, the optimization procedure has to be adapted in order to maintain safe operation throughout the experiment.

Sui et al. (2015) proposed SafeOpt, where safety constraints were added to the BO algorithm. Berkenkamp et al. (2020) later extended this method for multiple constraints with SafeOpt-MC. An alternative approach is Safe Active Learning (SAL) (Schreiter et al. (2015)) for control, where a GP model of the objective function depending on the controller parameters is actively learned by iteratively placing samples that maximize entropy. The model is then used in off-line optimization to find the optimal parameters. The design space is divided into safe and unsafe sets, and an additional discriminative model is trained to determine to which set a sample belongs. Safety constraints regarding the discriminative model are included in the active learning optimization problem, to probably avoid parameter regions that are known to lead to unsafe operation.

Schillinger et al. (2017) modified BO to include a discriminative model as a constraint in the acquisition function optimization. Their approach, Safe BO, was compared with standard SAL to tune a PI controller of a high pressure engine fuel supply system in a test vehicle. Safe BO required fewer function evaluation to converge to a better result, while SAL lead to higher exploration.

Khosravi et al. (2019) compared SAL with SafeOpt-MC to tune a heat pump digital PI controller based on a first principles model with parameters estimated using historical data. The tuning objective was to minimize both the overshoot and settling time, using the weighted sum method. Safety constraints were added to reflect concerns on using the procedure in the real process. Both algorithms converged to a solution identical to the one obtained using grid search,

although SafeOpt-MC displayed much faster convergence with less exploration.

After performing a literature review, it is noted that applications of surrogate-based optimization to tune PID controllers in chemical process case studies appears to be relatively unseen, in simulation experiments or otherwise. While testing the benefits of these methods, especially SafeOpt-MC or safe BO, as an alternative for model-free autotuning of industrial controllers is an interesting idea, there are still advantages to investigate their use in a simulation scenario. This is especially the case when testing preliminary control structures after process design, when simulations are the only source of data available.

4.1.2 Model Predictive Control (MPC)

Model Predictive Control (MPC) is an advanced control strategy that uses a process model to predict its future behaviour and determine the optimal sequence of control actions by solving an optimization problem on-line. The model can be either linear, usually in the form of a state-space matrix, or non-linear, in the case of Non-linear Model Predictive Control (NMPC). For more details the reader is referenced to reviews by Morari and Lee (1999) or Forbes et al. (2015) and the book by Rawlings et al. (2017). In addition to the quality of the prediction model, MPC performance of an MPC controller is dependent on the selection, or tuning, of several parameters. Beyond trial and error, methods to select these parameters are divided into heuristics, pole-placement or optimization, usually multi-objective (Alhajeri and Soroush (2020)). The latter is the focus of this section. A recent extensive review of tuning guidelines and methods for linear MPC controllers is provided in Alhajeri and Soroush (2020).

Because this tuning is a black-box optimization problem, without closed-form derivative expressions, similarly to PID tuning, it is usually solved using GA (Feng et al. (2018), Ramasamy et al. (2019)) or PSO (Nery Júnior et al. (2014)). The same motivation to use surrogate-based optimization in PID tuning also applies to MPC, however, it can be argued that using it for the latter offers more benefits. This is due to the manual effort associated with MPC tuning, lack of systematic approaches and especially since closed-loop simulations are more time consuming, because they require repeated solutions of an optimization problem. MPC design and tuning using computer simulation is usually a necessary pre-commissioning activity (Seborg et al. (2017)). Therefore, less trial and error and time spent in this stage leads to quicker deployment.

It is noted that BO has also been used in the field of robotics to solve black-box optimization problems in optimal control, Linear Quadratic Regulator (LQR), to optimize the weights (Marco et al. (2016)) and to actively learn the state-space model (Bansal et al. (2017)) based on closed-loop physical experiments on the real system. However, cases regarding MPC applications are discussed in more detail.

Instead of only optimizing the MPC parameters, similarly to Bansal et al. (2017), Piga et al.

(2019) propose using BO to derive the prediction model based on closed-loop performance. The rationale was that instead of treating system identification and control as two separate tasks, the process model should not necessarily be the one that minimizes a prediction error metric, but instead the one which provides the best closed-loop performance. A hierarchical control structure consisting of an inner loop digital PID controller and an outer loop MPC was considered. The approach was tested on a numerical example consisting of an inverted pendulum, optimizing the state-space model matrix entries, the prediction horizon and the PID controller gains. It was shown that the approach resulted in acceptable control performance with no previous knowledge of the system dynamics or prediction accuracy requirements.

Forgione et al. (2019) used GLIS to optimize the parameters of a linear MPC with embedded hardware to ensure real-time implementation constraints on a simulated inverted pendulum. The goal was to optimize the weights, control and prediction horizons, the state observer noise covariance matrices and soft constraints tolerances, for a total of 14 parameters. The method was successfully applied using two hardware platforms of different capability.

One considerable difficulty with using optimization methods to tune MPC controllers is to define the objective function to reflect the preference between different objectives. For multi-objective optimization this includes defining preferences a priori. This could be a considerable problem when the user does not possess extensive knowledge on MPC (Zhu et al. (2020)). The problem could be avoided by defining the preference by choosing a solution after approximating the Pareto set. However, this could be problematic when many objectives are present. Also, approximating this set could be impossible when performing tests on the real process.

Ramasamy et al. (2019) suggests iteratively cycling through weights when using the weighted sum method for multi-objective optimization. Their procedure uses an interactive decision tree, queuing the user at each iteration if closed-loop performance is improved. In their work, a GA was considered.

Zhu et al. (2020) proposed that instead of using a completely automated search relying on a single quantitative objective, a semi-automated approach based on qualitative pairwise comparisons of closed-loop responses could be used. The motivation is that often simply stating that response "A" is better than response "B" is easier to reflect user preference. Based on two close-loop responses, the user is queued to which one is better, and a GLIS based algorithm is used to suggest new combinations based on that classification, until a maximum number of function evaluations is met. Simulations on a simple chemical reactor case study showed the approach led to comparable performance with surrogate-based optimization using a quantitative objective function, defined by trial and error, while using a limited number of experiments.

4.2 Surrogate-based optimization for NMPC

Non-linear Model Predictive Control (NMPC) considers the use of a non-linear prediction model for MPC, which turns the on-line optimization problem into a complex non-convex NLP. Derivative-based algorithms are commonly used. However, these require gradient information, which can be unavailable or unreliable to estimate using finite differences in some cases. These include situations when the prediction model is not explicitly known, embedded in an external simulator or a variable-step ODE solver, is used for numerical simulation, thereby leading to numerical noise (Dæhlen et al. (2014)). This motivates the use of DFO methods. Their use for NMPC and when the prediction model is embedded in an external simulator can be categorized as a single-shooting method, because only the manipulated variables are parametrized (Dæhlen et al. (2014)). Surrogate-based optimization can be used to provide faster convergence than other model-free DFO in terms of objective function evaluations, and therefore better control performance. To the best of the author's knowledge, references about of surrogate-based optimization in NMPC are very scarce.

Marzat and Piet-Lahanier (2012) introduced a BO algorithm within an on-line NMPC framework for guidance control of a Unmanned Air Vehicle (UAV). EGO was used at each sampling time to determine the optimal control actions, based on an initial set of samples generated by LHD, until a maximum number of objective function evaluations is achieved. Because of this stopping criterion, the solution remained sub-optimal. BO could solve the optimization problem within the controller sampling time and was able to deal with non-quadratic and discontinuous cost functions.

Dæhlen et al. (2014) tested several trust-region methods against a derivative-based algorithm, SQP, in single-shooting NMPC for a subsea oil-gas separation process, described by highly non-linear, non-smooth model with stiff dynamics. Trust-region methods showed better closed-loop performance and higher robustness to numerical issues than SQP, which failed to converge when a variable-step ODE solver was used. A warm start strategy, where a part of the final model approximation in one iteration is kept the same for the next, also resulted in improved computational efficiency for one of the methods.

Yang et al. (2015) used two multi-objective BO algorithms, EHVI-EGO and (SMS-EGO within a NMPC framework to control a biogas plant, using a time consuming model for simulation. The algorithms were tested against an evolutionary algorithm (SMS-EMOA). The NMPC problem was defined as a minimization of two objectives, solved by approximating the Pareto front. EHVI-EGO slightly also out-performed the other algorithms and it was shown that the reference point value greatly influences the performance of the EHVI algorithm.

Negrellos-Ortiz et al. (2018) used a trust region method, BOBYQA, in the NMPC of an air separation unit. The prediction model was embedded in the Aspen Plus® Dynamics simulation environment, motivating a DFO approach. Different scenarios, including multivariate setpoint

changes, were tested. The method could be used effectively in NMPC when the prediction model is embedded in an external simulation environment. However, no comparisons with other methods were made.

4.3 Approximate Explicit MPC

In explicit-MPC, the control optimization problem is solved off-line for a high number of different system states and the solutions are stored in a look-up table (Bemporad et al. (2002)). In the on-line phase, the optimal control input is selected according to the current system state, avoiding the need for on-line optimization. This approach is more adequate for systems with fast-dynamics, where the optimal control problem must be solved fast. Surrogate models can be used to approximate the control law between system state and optimal control input. The advantages of using surrogate models is that an explicit model is used instead of the look-up table and when combined with DoE, a lower number of samples can be used to approximate the entire region of the system states, thus lowering the number of optimization problems to be solved off-line.

Shokry et al. (2016) compared the use of Ordinary Kriging (OK), Support Vector Regression (SVR) and Artificial Neural Networks (ANN) for explicit-MPC. Training and test samples were generated combining a Hammersley sequence and a fractional factorial design. Surrogate model performance was compared against on-line MPC in two case studies, showing a reduction from 78% up to 99% in computational time, with high accuracy and simplicity, avoiding the need to use a look-up table. OK showed the highest accuracy and flexibility, with the advantage of providing an estimated variance that can be used to assess the uncertainty of the control action.

Chakrabarty et al. (2017) combined SVM and sparse-grid interpolation for explicit-NMPC. A set of initial states was generated within the desired operating region using a low-discrepancy sequence and the MPC problem solved for each sample. A SVM classifier was used to approximate the feasible region boundaries, where constraints are not violated. Sparse-grid interpolation is then used to approximate the optimal control actions as a function of the system state within the feasible region, thereby approximately assuring that the control action does not violate constraints. The method was effectively applied to two case studies.

Chapter 5

Application 1: Real-time optimization using surrogate models

In this chapter, Surrogate-Based Optimization (SBO) is used for Real Time Optimization (RTO). Although RTO comprises several steps, including steady-state detection, data reconciliation, model parameter estimation and steady-state optimization (Seborg et al. (2017)), only the latter is considered in this chapter. The use of surrogate models for this task could be advantageous whenever the process model is embedded in an external simulation platform with no explicit model equations or derivative information. In addition to providing a simple benchmark to differentiate between optimization methods, this example serves the important objective of defining the optimum operating conditions for the subsequent case study, in the fashion of the previously mentioned hierarchic control structure.

5.1 Problem definition

To illustrate SBO approaches for RTO, the reactor from the well known Williams-Otto plant (Williams and Otto (1960)), widely used as a benchmark for RTO, (Forbes (1994), Navia et al. (2013), Singhal et al. (2016)), is considered as a case study. The reactor is an ideal CSTR with two pure reactant feed streams, F_A and F_B, and a single exit stream, F_R. In the reactor 3 parallel irreversible and exothermic reactions occur, all in the liquid phase:

$$\mathrm{A} + \mathrm{B} \longrightarrow \mathrm{C} \qquad (\text{Reaction 1}) \tag{5.1}$$

$$\mathrm{B} + \mathrm{C} \longrightarrow \mathrm{P} + \mathrm{E} \qquad (\text{Reaction 2}) \tag{5.2}$$

$$\mathrm{C} + \mathrm{P} \longrightarrow \mathrm{G} \qquad (\text{Reaction 3}) \tag{5.3}$$

In this problem, C is an intermediate compound, P and E are desired products and G is an undesired by-product. The reactor has two sets of coils, from which heat is removed or provided by water or steam, respectively, according to operational demands. As in Forbes (1994), several

simplifications are considered:

- Product prices and reactant costs remain constant
- Steady-state operation is considered
- F_A, is kept at a fixed value, 1,8275 kg/s
- Isothermal operation
- The only decision variables are F_B and the reactor temperature, T_R
- Only inequality constraints are bounds on manipulated variables
- The recycle stream is the original plant is set to zero
- The reactor liquid mass hold-up is kept at a constant amount
- Utility costs regarding cooling water or steam consumption are not considered
- No noise is considered, state variables are fully measured
- The model perfectly represents the plant, no plant-model mismatch exists

The detailed dynamic reactor model and parameter values are presented in Appendix A. The steady-state model used for this example was derived by neglecting the reactor mass, reactor temperature, and cooling water dynamics. The RTO objective is to maximize the economic profit, in $\$/s$ (Forbes (1994), Navia et al. (2013)):

$$\max_{F_B,T_R} Profit = 1143,38X_P(F_A + F_B) + 25,92X_E(F_A + F_B) - 76,23F_A - 114,34F_B \tag{5.4a}$$

s. t.

$$X_P = f(F_B, T_R) \tag{5.4b}$$

$$X_E = f(F_B, T_R) \tag{5.4c}$$

$$3 \leq F_B(kg/s) \leq 6 \tag{5.4d}$$

$$343,15 \leq T_R(K) \leq 373,15 \tag{5.4e}$$

X_E and X_P are the mass fractions of E and P, respectively. Despite being an NLP, in this operating range, the profit surface is strictly concave (Figure 5.1), with an unique local optimum, which makes it a useful case study for comparing different optimization approaches.

In this optimization problem the partial mass balance equations to E and P are the nonlinear equality constraints, which, in this case, are explicitly known. However, when considering black-box models, these equations are usually not available. To test SBO approaches, it is considered that these represent unknown constraints. To solve this black-box optimization problem a surrogate model could be used to approximate each of the two equality constraints and the profit would be a function of the output of these models. With the exception of ANN, this would require fitting two models. Another option is to approximate directly the profit as a function of the input variables, as shown in Figure 5.2. The latter option was considered more efficient for the purpose of optimization, without the need to fit several surrogate models.

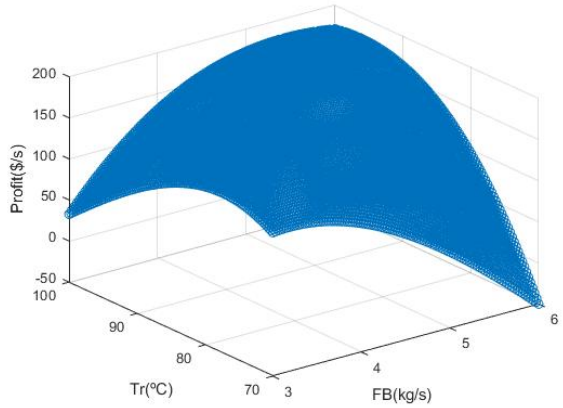


Figure 5.1: Profit surface of the Williams-Otto reactor

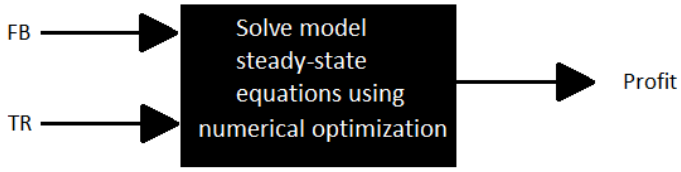


Figure 5.2: Diagram for steady-state optimization

5.2 Model selection and validation

The objective of surrogate based optimization is to find the optimum with the minimum number of objective function evaluations. So the first step considered in this case study is the development of a surrogate model that can accurately approximate the objective function in the operational range with the least amount of training points needed. The models considered are polynomial regression, Kriging, ANN, RBF and SVR.LHD, Sobol and Halton sequences are used to generate the training data. All models and sampling strategies are implemented using the Statistics and Machine Learning Toolbox™ and the Deep Learning Toolbox™ from MATLAB®. The reactor steady-state values are determined by solving the system of algebraic non linear equations using numerical optimization, trough the MATLAB® function *fsolve*. In this analysis, several factors are changed for comparison:

- Surrogate model hyper-parameters (kernel on kriging and SVR, number of layers and nodes in ANN, etc)
- Initial sample size
- Initial DoE scheme

Training data consisted of samples with three different sizes: 20 ($10D$), 30 and 50 points.

LHD designs are obtained using *lhsdesign*, Sobol designs using *sobolset* and Halton designs using *haltonset*. In quasi-random designs, for simplicity sake, the first values of the sequence were considered, so that no points were skipped. Due to the randomness of LHD, each sample size was generated 10 times and the median of the prediction RMSE was considered representative.

The test data consisted of a 10201 point (101^2) regular grid (Figure 5.1). Usually, if using this number of test points was viable, there was no motivation to use a surrogate model. However, since the objective function in this case is cheap to evaluate, the approach was considered useful to thoroughly evaluate each model ability for global approximation. Because the testing sample remains the same for all models, the prediction error is not normalized.

It must be noted that the hyper-parameter selection for these models can be done automatically, for example, formulating an optimization considering the minimization of the CV error, where the decision variables are the model hyper-parameters. However, in this work, these were selected based on a visual inspection of the prediction error. Details regarding this selection are presented in Appendix B.

5.2.1 Comparison between different initial sampling designs

After selecting the surrogate model hyper-parameters (cf. Appendix B), the impact of different sampling designs on model approximation is assessed. In each of those cases, because the model which performed better than others across all sampling designs was chosen, a biased choice towards a sampling design can be neglected. With this consideration, the same data used before can be used to determine the best design plan, which is considered to be the one which leads to a consistently lower error for most models.

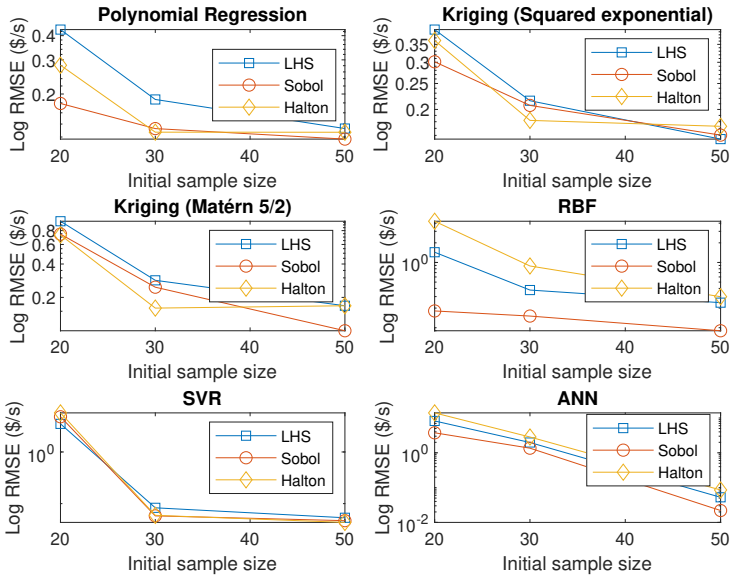


Figure 5.3: Comparison between different initial sampling designs (logarithmic scale used to ease comparison between plans)

Observing Figure 5.3, it becomes apparent that despite existing a gap between the performances of different designs for the smallest initial sample, that gap diminishes when the size is increased. This could mean that beyond a certain point, for a large number of samples, differences in performance between design plans could be relatively low. Regarding different designs, it can be seen that the Sobol design leads to lower error when using either the smallest and largest sample size for almost all cases. For these reasons, this design is considered the one with the best performance.

5.2.2 Comparison between different surrogate models

Comparing between surrogate models, it can be seen that for when using a smaller amount of data, a polynomial regression model generally shows the best accuracy, followed by OK with the squared exponential kernel. However, RBF show very good accuracy with a lower sample size when a Sobol design is used. When a larger sample is used, SVR and Artificial Neural Networks (ANN) stand out as the most accurate models. Considering the best results of an initial Sobol design with 50 samples, the prediction error for all models is compared in Table 5.1. Because time required to build and evaluate a surrogate model is also of importance, especially if that model is to be used on-line, such as in RTO, fitting and prediction times are also compared. These times are obtained repeating the fitting and prediction process 15 times for each model and taking the median value.

Table 5.1: Comparison between performance of different surrogates for an initial Sobol design with 50 samples

Model	RMSE validation ($\$/s$)	Fitting time (s)	Prediction time (s)
Polynomial Regression	0,1171	0,0118	0,0051
Kriging (Matérn 5/2)	0,1001	0,0228	0,0080
ANN	0,0220	1,8500	0,0108
RBF	0,0898	0,0223	0,0133
SVR	0,0468	1,6098	0,0016

*Run on an Intel Core(TM) i7-8550U @1.8 GHz, 8GB RAM, Windows 10 64 bits

The polynomial regression model shows the lowest accuracy. However, because of its simplicity, the expectation was that it performed much worse than other models. This unexpected performance can be due to the fact that this problem is low dimensional, with only two design variables, and the underlying model of the objective function, despite non-linear, does not have peaks and valleys, i.e., is convex. Despite the lower accuracy, the training time is the lowest of all models, due to parameters being estimated using OLS. Prediction times are also low.

RBF models, despite not having the highest prediction accuracy for this sample size, consistently show a low prediction error across lower sample sizes, while having low fitting times.

The use of a parametric Gaussian kernel also allowed some flexibility and enabled better predictions.

Kriging models despite not being as accurate, are also relatively quick to train and offer a measure of confidence in prediction by also providing the predicted variance, which can be used to make a decision about the model quality.

ANN shows by far the highest accuracy, followed by SVR. However, this comes at the cost of very large training times, almost two orders of magnitude greater than the remaining models. This is because the fitting process requires solving a highly non-linear optimization problem with many local minima for ANN and a constrained quadratic optimization problem in the case of SVR. Despite the high training times, predictions with a SVR model are extremely fast, even faster than a polynomial model. ANN requires a larger sample size, which may be prohibitive for more expensive simulations, the fitting times are very large. Also, despite the fact that different architectures provide a higher flexibility for ANN, because of the high number of possibilities, manually selecting the best one is a difficult task. However, automatic procedures such as the ones mentioned in chapter 2.4.3 could ease this selection.

SVR models also requires more training data and fitting time, but are accurate and fast to predict, making them useful when cheap data is available, training time is not a problem and fast and accurate predictions are needed.

5.3 Surrogate-based optimization using static sampling

In this section, the surrogate models built with a Sobol design of 50 samples are used for optimization. These models are optimized using derivative-based and DFO methods. Because the assumption is that the surrogate models can be cheaply evaluated, no limit was imposed on the number of evaluations of these functions, keeping the stopping criteria for all the solvers at default values. Because the surrogate models are an approximation of the objective function, the predicted optimum values need to be validated on the original objective function, so an extra evaluation was needed.

To assess the surrogate optimization approach using static sampling, these models are compared with other popular optimization algorithms in MATLAB®, including a local search gradient based algorithm using Sequential Quadratic Programming (SQP) (*fmincon*), a local search DFO algorithm that uses the Nelder-Mead Simplex (*fminsearch*), and a stochastic DFO Genetic Algorithm (GA) (*ga*). An initial guess of 4 kg/s and 353,15 K and a budget of 51 total objective function evaluations was supplied for both *fmincon* and *fminsearch*. For *ga*, only one generation with 50 individuals, resulting in 50 total function evaluations, was used. For convenience, the optimization problem was solved as a minimization problem, minimizing the negative of Equation (5.4a). The results are presented in Table 5.2.

Table 5.2: Optimum profit value ($\$/s$) obtained using SBO and static sampling

Algorithm	Black-box model	Polynomial	Kriging	SVR	ANN	RBF
fmincon (SQP)	190,9906	190,9906	190,9906	190,9906	190,9906	186,1710
fminsearch	190,9551	190,9906	190,9906	190,9906	190,9906	190,9905
GA	190,7142	190,9906	190,9906	190,9906	190,9906	190,9906

For optimization algorithms without surrogate models, only *fmincon* was able to find the optimum. This could mean that even if closed-form derivative expressions are not available, but the objective function is fast to evaluate and smooth, so that finite-differences can be used, derivative-based algorithms can still be of use. The result of GA was expected because this algorithm usually requires many objective function evaluations to converge.

Regarding the use of surrogate models, it can be seen that, with exception for the RBF, all algorithms find the optimum within a tolerance of $1 \times 10^{-5} \$/s$. Also, differences between optimizing surrogate models with different algorithms appear to be negligible, except for the RBF model, which does not appear to handle derivative-based algorithm, such as *fmincon*.

5.4 Surrogate-based optimization using adaptive sampling

The optimization based strategies discussed in chapter 3, MSRS and BO are used to compare the performance of static and adaptive sampling. A maximum number of 51 objective function evaluations was considered for both methods, using two initial samples of different sizes, $5D$, and $10D$. These were generated with a Sobol sequence. BO was used with the EI criteria. The convergence plots are shown in Figure 5.4.

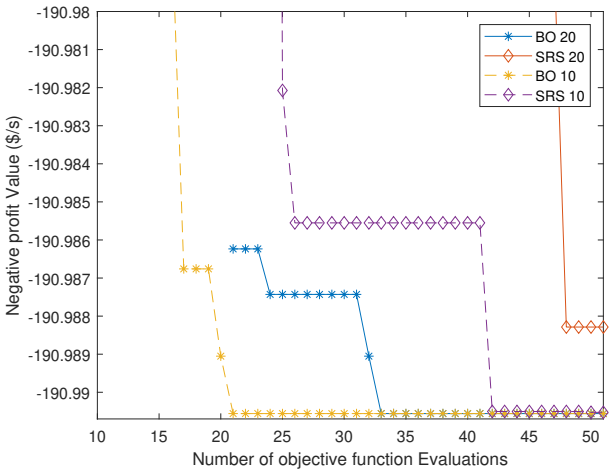


Figure 5.4: Bayesian optimization and SRS convergence plot (Negative profit value shown due to minimization)

BO shows better performance than MSRS, which does not find the optimum. The use of a smaller initial sample seems to increase convergence, specially in BO. This could mean

that is it better to begin with a lower sample size and a lower model approximation and spend more function evaluations on adaptive points. However, this remains to be assessed on a more difficult, non-convex, problem.

BO found the optimum within a tolerance of 1×10^{-5}, the same as when using surrogate models and static sampling. However, it is important to state that only 21 total function evaluations were required, less than half the ones used in static sampling. In this case, 50 samples clearly represent oversampling. This illustrates the difficulty in determining the training sample size when using static sampling approaches. Moreover, the hyper-parameter selection is also not an easy task, especially for ANN. Therefore, the tasks of selecting a surrogate model, sampling plan and sample size require additional time and effort, which can be prohibitive when considering the optimization is to be performed on-line, as is the case in RTO. Thus, SBO using adaptive sampling is considered more efficient for lower dimensional RTO problems.

5.5 Optimization considering plant disturbances

In the previous sections, the optimal operating point was determined for a nominal operation scenario, without disturbances. In this section, the optimization problem of Equation (5.4a) is solved considering measured disturbances, which take the form of step changes with a magnitude of $0,2647 kg/s$ to F_A, that can occur approximately every 20 minutes (Forbes (1994)). All other details are kept the same as before. The optimization problem is solved with the algorithms that displayed better performance, BO with 10 initial samples generated using a Sobol sequence and *fmincon*. A limit of 50 objective function evaluations was imposed for both.

Table 5.3: Optimum profit and input variable values for each scenario

Algorithm	$F_A(kg/s)$	**Profit** ($\$/s$)	$F_B(kg/s)$	$T_R(K)$
BO	1,5628	177,8999	4,1605	361,0741
	1,8275	190,9906	4,7893	362,8608
	2,0922	202,0214	5,3952	364,3278
fmincon (SQP)	1,5628	177,8999	4,1624	361,0930
	1,8275	190,9906	4,7875	362,8528
	2,0922	202,0217	5,3979	364,3689

Contrary to BO, *fmincon* was able to find the optimum for all three scenarios, achieving convergence within the function evaluation budget. This confirms that when the objective function is smooth, convex and derivative information cheap to obtain, derivative-based methods are efficient. However, these assumptions usually do not apply to time consuming black-box functions. Also, in the single case when BO finds a worse optimum than *fmincon*, the difference is negligible. Nonetheless, the optimum F_B and T_R values obtained with *fmincon* are considered better and used to determine the optimum operating points the next case study.

Chapter 6

Application 2: Optimal PID tuning using surrogate-based optimization

In the previous chapter, optimization using different surrogate models and sampling strategies was applied to a RTO problem. In that example, the objective function to be approximated was simple, fast to evaluate and with known and convex shape. In this chapter a more complex problem is considered, which involves tuning two PID controllers trough the use of simulation based optimization, as introduced in chapter 4. Contrary to the previous application, the underlying model between controller parameters and quantitative metrics based on the system's closed-loop response is a true black-box model, with unknown closed-form. Additionally, simulation experiments are more time consuming since each run involves solving a large system of non-linear ODEs and the need to link MATLAB® to the simulation model. SBO, using the adaptive sampling methods described in chapter 3, are used to solve this black-box optimization problem and compared with other DFO approaches and traditional PID tuning methods. MATLAB® and Simulink® are used to perform the closed-loop simulations and optimization.

6.1 Williams-Otto reactor control structure and objectives

The Williams Otto reactor (Williams and Otto (1960)) is again considered as a case study, using a first principles model to simulate the system dynamic behaviour. The original model presented in Williams and Otto (1960) was modified to include a mass balance to the reactor and an energy balance to the cooling water, described by a PDE. After discretization, this dynamic model is composed of 208 ODEs, 208 state variables, 7 input variables and 12 parameters. A simplified Piping and Instrumentation Diagram (PI&D) of the reactor, along with details regarding model development, modelling assumptions and input variables and model parameters values are presented in Appendix A. This model is implemented in Simulink® using a MATLAB® function block (Appendix C).

The pairings of control and manipulated variables are kept as the ones originally described

by Williams and Otto (1960), with the exception of the heating coils, which are disregarded in this work. The tuning of the F_B, flow controller is also not considered. The reactor liquid mixture level, L_R, is controlled by manipulating the reactor exit stream volumetric flowrate, Q_R, and T_R is controlled though the manipulation of cooling water entry volumetric flowrate, Q_{cin}. Both controlled variables are under feedback control, represented by a simplified block diagram for a single controlled variable in Figure 6.1.

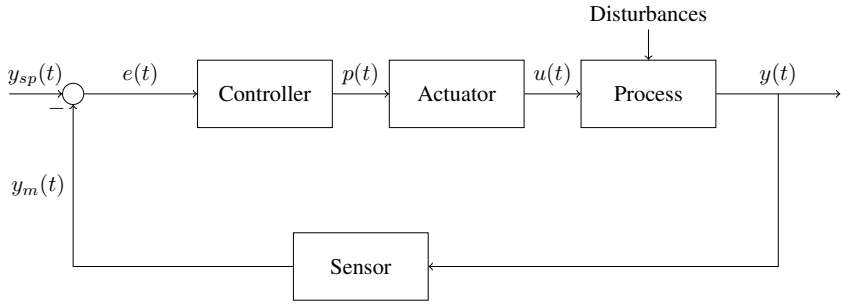


Figure 6.1: Block diagram of a single loop negative feedback control structure.

In the diagram above, y_{sp} is the controlled variable desired value, or setpoint, y_m is the measured value of this variable, e the difference between these two values, or error, y the process output, u the process input and p the controller output, or signal.

In this case study, a simplified scenario is considered where the dynamics of both the sensor and final control element are considered negligible. Also, the models of these components are assumed to be described by a unitary gain. With these considerations in mind, the notation y_m=y and p=u is used from here on out. Additionally, it is considered that the controlled variables signal is perfectly filtered, so no measurement noise is present, and that measurements are available in continuous time. However, the control element is a digital, discrete-time controller, where feedback information and control action is received and sent, respectively, at a pre-determined time interval, or sampling time, Δt. It must be emphasized that this value regards the sample period used in the control calculations, and that process data can be sampled at faster rates for monitoring purposes. This example considers the use of a Proportional integral derivative (PID) controller with the expanded, or non-interacting form ((Seborg et al. (2017))) of the control algorithm:

$$u(t) = \bar{u} + K_P e(t) + K_I \int_0^t e(t^*)dt^* + K_D \frac{de(t)}{dt} \tag{6.1}$$

Where $\bar{u}$ is the steady-state value of the control signal, or bias, K_P the proportional gain, K_I the integral gain and K_D the derivative gain. Since each gain independently influences only one control mode, this controller form is well suited for optimization-based tuning approaches. The corresponding discrete-time transfer function version of Equation (6.1) can be derived using a z-transform and considering a backwards Euler formula, which usually leads to higher stability,

for the integrator and derivative terms (Seborg et al. (2017):

$$\frac{u(z)}{e(z)} = K_P + K_I \Delta t \frac{z}{z-1} + \frac{K_D}{\Delta t}\frac{z-1}{z} \tag{6.2}$$

After manipulation, the controller output at each time interval k is given by:

$$u(k) - u(k-1) = K_P(e(k) - e(k-1)) + K_I \Delta t e(k) + \frac{K_D}{\Delta t}(e(k) - 2e(k-1) + e(k-2)) \tag{6.3}$$

Clamping, where upper and lower limits are imposed on the rate of control action ($u(k) - u(k-1)$), is added to prevent overrange and controller output saturation. The controller is implemented using the PID controller block in Simulink® (Appendix C).

Controller performance depends on a careful and thoughtful selection of both the controller gain and sampling time values, in a procedure known as controller tuning. While much discussion and a variety of methods exist to select controllers gains, the selection of a controller sampling time remains more of an art than science (Seborg et al. (2017)). The exact choice of this parameter depends on several factors such as the dynamic characteristics of the system and available hardware. While a high sampling time leads to loss of information and deterioration of control performance (Seborg et al. (2017)), a very low sampling time may increase computational requirements without increasing control performance. One guideline regarding the process dominant time constant, τ, is recommended by Åström and Wittenmark (2011):

$$0,01 \leq \frac{\Delta t}{\tau} \leq 0,05 \tag{6.4}$$

The introduction of an analogue-to-digital converter, the controller sampler, adds an effective delay to the system response. In order to use available tuning relations Seborg et al. (2017) recommend adding half the sampling time value to the process model time delay. A properly tuned controller should ideally:

- Provide closed-loop stability to the process
- Enable fast and accurate setpoint tracking, with minimal or no steady-state error
- Reject the effect of disturbances on the controlled variable
- Avoid excessive control action
- Be robust to modelling errors (plant-model mismatch) and variable operating conditions

Controller tuning is a compromise between performance and robustness (Seborg et al. (2017)). After tuning a controller, its performance can be assessed trough a variety of quantitative metrics, such as the steady-state error, or offset, integral error, overshoot, settling and rise time. Qualitative information regarding the graphical analysis of the closed-loop system response, for instance, the amount of oscillation, is also important.

The control objectives are to maintain as close to economically optimal operation as possible, considering a simulation scenario with a time window of 60 minutes. The process variables are initially on the nominal operating point and two step changes in F_A occur, one to 1,5628 kg/s and the other to 2,0922 kg/s, after 20 and 40 minutes of simulation, respectively. Because of these disturbances, the setpoints of the controlled variables are no longer optimal, and new ones must be calculated. In traditional steady-state RTO, new setpoints would only be recalculated once the system is considered to be at its previous steady-state (or approximately), stabilized by the regulatory controllers after the disturbance occurs (Seborg et al. (2017), Straus (2018)). However, to simplify the analysis, it is considered that the optimal setpoints for each value of F_A have been previously computed off-line, in an open-loop manner, considering the optima obtained with *fmincon* in Chapter 5. Therefore, no wait for a return to steady state is necessary, and new setpoints are sent to the controllers immediately once the disturbance in F_A is measured. Assuming that the F_B flow controller is well tuned and that the typical response of these loops is in the order of seconds (Seborg et al. (2017)), it is also considered that this controller moves F_B to its setpoint instantaneously and with perfect control, once F_A changes. Since the steady-state optimization was performed considering a constant mass holdup in the reactor, the level setpoint remains constant. The optimum operating points are presented again in Table 6.1. Because component mass fractions are not directly controlled variables, their values are not shown.

Table 6.1: Operating points considered in closed-loop simulations

Variable	Nominal operating point	Operating point 1	Operating point 2
$F_A(kg/s)$	1,8275	1,5628	2,0922
$F_B(kg/s)$	4,7875	4,1624	5,3979
$T_{Rsp}(K)$	362,8528	361,0930	364,3689
$L_{Rsp}(m)$	2,3897	2,3897	2,3897
Profit ($\$/s$)	190,9906	177,8999	202,0217

Achieving close to optimal economic operation requires that both the level and temperature controllers are properly tuned. In this work, the tuning procedure is treated as an optimization problem. While the level controller is tuned for disturbance rejection, the temperature controller is tuned to provide both setpoint tracking and disturbance rejection. In this regard, common metrics to assess controller performance in the time domain, namely, overshoot, rise or settling time were not considered to be adequate for direct optimization. In lieu of these metrics, the Integral of the Squared Error (ISE), which penalizes large deviations from setpoint (Seborg et al. (2017)) is considered:

$$\int_0^{t_f} (e(t))^2 dt = \int_0^{t_f} (y(t) - y_{sp}(t))^2 dt \tag{6.5}$$

This integral is approximated numerically using the trapezoidal rule. Another important tuning objective is to reduce control effort, so the controller output is smooth. This serves the important task of reducing wear and tear on the final control element and can be quantified as:

$$\sum_{k=1}^{N} \mid \Delta u_k \mid = \sum_{k=1}^{N} \mid u_k - u_{k-1} \mid \tag{6.6}$$

Where N is the total number of control actions. The optimization objectives are made dimensionless:

$$f_1 = \int_0^{t_f} \left(\frac{y(t) - y_{sp}(t)}{y_{sp}(t)} \right)^2 dt \tag{6.7}$$

$$f_2 = \sum_{k=1}^{N} \frac{\mid u_k - u_{k-1} \mid}{u_{max} - u_{min}} \tag{6.8}$$

The controller tuning problem is formulated as a box-constrained bi-objective optimization problem, by minimizing both f_1 and f_2 simultaneously. As described in Chapter 3, this problem is transformed into a single-objective problem by the weighted sum method:

$$\min_x \ [f_1(x), f_2(x)] = \ w_1 \int_0^{t_f} \left(\frac{y(t) - y_{sp}(t)}{y_{sp}(t)} \right)^2 dt + w_2 \sum_{k=1}^{N} \frac{\mid u_k - u_{k-1} \mid}{u_{max} - u_{min}} \tag{6.9a}$$

s.t.

$$y(t) = f(x, t) \tag{6.9b}$$

$$u_k = f(x, t) \tag{6.9c}$$

$$x_{lb} \leq x \leq x_{ub} \tag{6.9d}$$

$$x = [K_P, K_I, K_D, \Delta t], \ x \in \mathbb{R}^4 \tag{6.9e}$$

Where t_f is the total simulation time, and x is the vector of controller parameters to be optimized, which includes its gains and Δt. Commonly, this last parameter is set to a fixed value before solving the optimization problem. However, since the optimum controller gains depend on the sampling time (Laskawski and Wcislik (2016)), choosing this value before performing the optimization could require some trial and error. In addition, if closed-loop performance when using the optimum controller gains proves to be undesirable, the optimization problem would have to be solved for a different Δt, which would require re-defining bounds on the controller gains. Thus, to reduce trial and error, Δt is treated as an additional continuous decision variable. w_1 and w_2 are the weights assigned to the first and second objective, according to their relative importance. Equation (6.9b) and Equation (6.9c) are the equality constraints with unknown closed-form. It is also noteworthy that the use of an absolute operator introduces a discontinuity in the objective function, which favours the use of DFO methods.

It must be emphasized that these metrics represent the authors choice, and are only two in a multitude of different ones that can be considered. The advantage of using an optimization approach for controller tuning is that the objective can be changed to reflect the preference of the decision maker. For instance, if the controller is required to have fast setpoint tracking, the rise-time could be used as one objective to minimize. Another example could be to optimize controller performance, while also reducing utility consumption.

Standard model-based and model-free PID tuning methods are used to compare the performance of SBO. The former category includes tuning relations based on IMC methods, optimization of an error criteria and other miscellaneous relations. The latter includes the Ziegler-Nichols (Z-N) (Ziegler and Nichols (1942)) and Tyreus-Luyben (T-L) (Tyreus and Luyben (1992)) tuning relations, based on the ultimate gain, K_{cu} and the ultimate period, P_u. These last values are obtained using the Relay autotuning method (Åström and Hägglund (1984)), in which the PID controller is temporarily switched to an on-off controller, or relay, with the following output:

$$u(t) = \begin{cases} d & , \text{ if } e(t) \geq T \\ -d & , \text{ if } e(t) < -T \end{cases} \tag{6.10}$$

A positive threshold value, T, is added to the error to prevent excessive switching due to high frequency measurement noise. This is known as a relay with hysteresis (Åström and Hägglund (1984), Yu (2006)). In this example, no noise is considered, T is set to zero, and the relay is ideal. After the switch to on-off control, the controlled variable exhibits a response with a characteristic sustained oscillation. Based on this response, P_u is determined as the time interval between the highest oscillation peaks and K_{cu} can be approximated as (Åström and Hägglund (1984)):

$$K_{cu} = \frac{4d}{\pi a} \tag{6.11}$$

Where d is the amplitude of the controller, or relay, output and a the amplitude of the controlled variable oscillation. The relay tests are implemented using the relay block in Simulink® (Appendix C). The Relay-feedback test will also be used for closed-loop system identification, a procedure described further in section 4. Due to the fact that controller tuning relations used in this work are usually presented for an ideal version of a PID controller, when necessary, the following expressions are used:

$$K_P = K_c \tag{6.12}$$

$$K_I = \frac{K_c}{\tau_I} \tag{6.13}$$

$$K_D = K_c \tau_D \tag{6.14}$$

All closed-loop simulations are implemented in Simulink®, solving the system of ODEs with the variable-step MATLAB® solver, *ode45*, based on a explicit Runge-Kutta formula, with

a relative tolerance of 10^{-6} and a maximum step size of 0,1 seconds. A simplified description of the MATLAB® function used to run and extract information from Simulink® simulations is presented in Appendix C. That function is then called by an external MATLAB® optimizer, in this case, *bayesopt*, *surrogateopt* or *ga*.

6.2 Proposed methodology for simulation optimization-based tuning

In addition to the selected optimization algorithms, the success of the controller tuning approach based on the optimization of Equation (6.9a), depends on two critical factors: the definition of lower and upper bounds of the decision variables, or design space, and the selection of the objective function weights. To reduce trial and error, a simple systematic methodology is presented.

The definition of the design space should include system knowledge. The gains of a digital controller can be obtained by using tuning relations for different Δt. This last parameter can also be defined using heuristics related to the process dynamics, such as the dominant time constant, in the desired operating region. It becomes apparent that a process model, in the form of a low order transfer function, is valuable to aid in this task. In addition, a model can indicate a need to select different controller structures, for example, if the use of derivative action is needed. If not available, this model can be obtained using two different strategies: by linearisation of the non-linear dynamic model or through system identification using dynamic simulations.

The first strategy is straight-forward if the non-linear model structure is known and of relatively low order. If the model is high-order, a lower order version can be obtained using model-based order reduction, for instance balanced truncation (see Gugercin and Antoulas (2004)). However, this can prove difficult if there are a high number of unstable states or the model is of very high dimension. In addition, an explicit model is required, which may not be always available for some situations. An alternative is to use system identification techniques based on a simulated response.

If the system is open-loop stable, model identification can be easily accomplished by performing a step test. However, if the system is open-loop unstable, closed-loop identification techniques such as the ones mentioned in chapter 4 are required.

It is important to state that the identification procedure goal should not be to obtain a process model with a high degree of accuracy, because a non-linear process model already exists. Instead, the goal is to obtain an approximate model only to get a notion of the design space range so that optimization can lead to better results with fewer function evaluations and time. Moreover, the ultimate goal of using simulation optimization is to automate the controller tuning procedure. Therefore, more complex identification procedures requiring several tests, more

computations or control theory knowledge, should in principle be avoided. With this in mind, a single experimental test should be sufficient to obtain the model and in this thesis, a relay test, which requires fewer computations, is recommended over the closed-loop step test. Because the system is open-loop unstable and the identification procedure requires an initial steady-state, a controller must be previously specified and because the system is unknown, choosing the controller parameters some requires manual. A simple Proportional or PI controller is sufficient for this task. In this regard, the approach is not completely void of trial and error in some cases.

The identification procedures, both open and closed-loop should be performed without in continuous time. Additionally, if confidence in the process model accuracy is low, controller tuning relations that incorporate robustness are recommended and the design space limits should be extended.

The step of system identification can be avoided either if a controller is already specified in simulation and needs to be retuned or if a continuous time controller is to be tuned. In this last case, a relay test followed by Z-N type tuning relations can be used to define the design space.

The objective function weights must be selected to reflect the decision maker preferences. If the objectives scales are of different magnitudes, the weights can mask a scaling problem and compromise the attribution of preferences (Marler and Arora (2010)). Therefore, the objectives should be made of similar scale before assigning values for the weights. This can be a difficult task if the range of values that these objectives take is not known a priori, as is the case in dynamic simulations. A simple solution is to use the initial sample obtained through DoE, required to initialize the SBO procedure, to obtain a distribution of the different objectives values in the chosen design space. Because the designs are inherently space-filling, they should, in principle, serve to obtain a representative distribution. In the case of two objectives, the ratio between these values can be calculated for each sample. The median of this ratio distribution can be used as the weight for one of the objectives. The scales of the two objectives should become similar and the other weight can be used to reflect the decision maker preference. Summarizing, the proposed approach consists of the following steps:

1. Define control objectives, performance metrics and simulation scenario ;
2. If not available, obtain a transfer function model either by linearisation or system identification ;
3. Use heuristics to define the controller sampling time range and add half that value to the process model delay ;
4. Use tuning relations to obtain controller gains for different sampling times and perform initial simulations ;
5. Define the design space based on the information provided by the last step ;
6. Generate initial samples with DoE to obtain a distribution of the different objectives values ;
7. Use a statistical measure (e.g. median) to define the objective scales ;

8. Assign weights that reflect preferences ;
9. Solve the optimization problem using Surrogate-Based Optimization and the initial samples generated by DoE.

6.3 Level PI controller tuning

The liquid level controller tuning serves as a simple example to illustrate the proposed approach. This process transfer function can be easily deduced analytically by setting the variables in Equation (1) to deviation variables, considering a steady state operating point, taking the Laplace transform and considering Q_R as the manipulated variable:

$$L_R{}'(s) = \frac{-1}{A_R s} Q_R{}'(s) \Leftrightarrow \frac{L_R{}'(s)}{Q_R{}'(s)} = \frac{K}{s} \tag{6.15}$$

Where K is the process gain and takes the value of -0,8901 $(m.s)/m^3$. Processes described by a model of this type are known as integrating processes and present several unusual closed-loop characteristics. In addition to not being self-regulating processes, an increase in controller gain can reduce oscillation, instead of increasing it (Seborg et al. (2017)). As previously mentioned, half the value of Δt is added to process model, changing it to a Integrator-plus-time-delay (IPTD) model:

$$\frac{L_R{}'(s)}{Q_R'(s)} = \frac{K}{s} e^{-\frac{\Delta t}{2} s} \tag{6.16}$$

Guidelines to determine a suitable range for Δt based on the process time constant, for instance Equation (6.4), are not adequate for pure integrating processes, such as this one. Thus, as a base example, this parameter is assumed to be between 6 and 120 seconds. Because no disturbances to the level occur until 20 minutes of simulation and disturbance rejection proved to be relatively fast, all level closed-loop simulations were performed considering only 30 minutes of operation, instead of 60. Because the reactor mass needs to remain at a constant value, and deviations such as steady-state offset are not wanted, a Proportional Integral (PI) controller is considered for the control of the reactor liquid level.

6.3.1 Standard model-based and model-free methods

Three different model-based methods were used to tune the level controller, including the SIMC method (Skogestad (2003)), the minimization of the ITAE criterion for disturbance rejection (Poulin and Pomerleau (1996)) and the AMIGO method (Åström and Hägglund (2006)). The controller tuning relations are presented in Table 6.2.

Table 6.2: Tuning relations for a PI controller and IPTD process model

Method	K_c	τ_I
SIMC	$\frac{1}{K(\tau_c+\theta)}$	$4(\tau_c+\theta)$
ITAE (disturbance)	$\frac{0,5264}{K\theta}$	$4,5804\theta$
AMIGO	$\frac{0,35}{K\theta}$	$13,4\theta$
Tyreus-Luyben*	$0,31K_{cu}$	$2,2P_u$
Ziegler-Nichols*	$0,45K_{cu}$	$\frac{P_u}{1,2}$

*General, not based on any process model

Closed-loop simulations were performed considering these relations and several values of Δt (Appendix E). As the value of this parameter increases, controller performance degrades, as expected. Considering these results, a Δt of 6 seconds is chosen. When implementing a Relay-feedback test using a digital controller, the experiment sampling time should be as low as possible, to enable accurate identification (Berner et al. (2016)). Therefore, a Δt of 6 seconds was also chosen for the test. Also, no disturbances are considered during this test. The test was implemented using a Relay block and a zero-order hold block in Simulink® (Appendix C). Considering a relay amplitude of 3,741 $\times 10^{-3} m^3/s$, the results are shown on Figure 6.2.

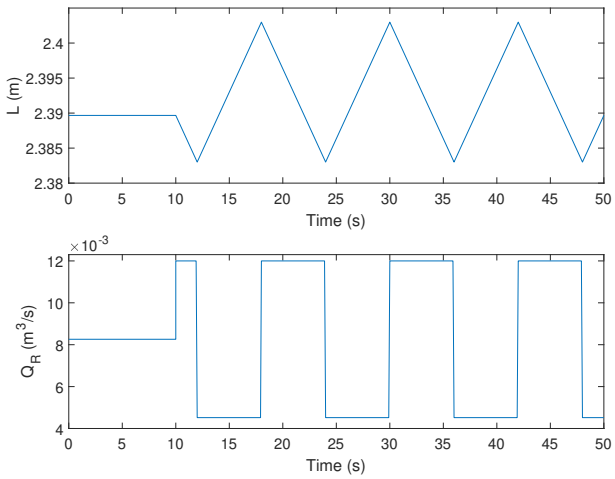


Figure 6.2: Level Relay test

Based on the response in Figure 6.2, a is approximately 0,01 m and P_u is 12 s. For these values, Equation (6.11) gives a K_{cu} of 0,4763 $m/(m^3/s)$. A negative controller gain is considered to ensure negative feedback.

6.3.2 Comparison between standard tuning and optimization-based tuning

In the optimization-based tuning approach the performance of both BO and MSRS was compared with that of GA. A maximum budget of 100 total objective function evaluations was imposed. For BO the EI criteria was chosen. GA used 4 generations with a population of 25 individuals. Because all the traditional tuning methods displayed competitive performance, the design space was considered to be contained inside the domain defined by the maximum and minimum values of the controller gains for the various tuning relations, for sample times of 6 and 30 seconds:

$$6 \leq \Delta t(s) \leq 60 \tag{6.17}$$

$$-0,22 \leq K_P \leq -0,026 \tag{6.18}$$

$$-0,022 \leq K_I \leq -1,3 \times 10^{-4} \tag{6.19}$$

For SBO, 30 initial samples ($10D$) were generated within the variable bounds above using a Sobol design. As previously mentioned, these samples are used to obtain a notion of the magnitude between the two objectives. From this sample, the median of the ratio between f_1 and f_2 was approximately 0,0065. Setting w_2 to 0,0065, the objectives should present approximately similar scale. w_1 then is chosen to be 10, which means that control performance is considered to be approximately 10 times more important than smooth control action. It must be noted that since these weights are chosen a priori, their values may not reflect this exact relative importance. For all algorithms, the random number generator was set to default, for reproducibility.

Table 6.3: Level controller tuning optimization results for 100 total function evaluations

Algorithm	**Objective function value** ($\times 10^3$)	**Time (s)**
GA	8,8373	184,14
SRS	5,8278	171,45
BO	5,0481	226,99

*Run on an Intel Core(TM) i7-8550U @1.8 GHz, 8GB RAM, Windows 10 64 bits

For a fixed number of function evaluations, GA showed the worst performance and BO the best, followed by MSRS. BO required more time than the other algorithms due to the auxiliary acquisition function optimization problem at each iteration. Nonetheless, it was able to find the lowest objective function value given the same evaluation budget. The controller parameters obtained for different methods are shown in Table 6.4 and the respective closed-loop simulations in Figure 6.3. To enable a quantitative comparison, the two metrics considered as optimization objectives and the normalized Integral of the Absolute Error (IAE) for each closed loop simulation are are also shown in Figure 6.4.

Table 6.4: Parameters for reactor level PI controller

Method	Δt **(s)**	K_P	K_I ($\times 10^2$)
SIMC ($\tau_c = \theta$)	6,0000	-0,1872	-0,7802
ITAE (disturbance)	6,0000	-0,1971	-1,4346
AMIGO	6,0000	-0,1311	-0,3360
T-L	6,0000	-0,1477	-0,5593
Z-N	6,0000	-0,2143	-2,1433
BO (EI)	6,0177	-0,2184	-1,2501
SRS	6,0462	-0,1749	-0,9159
GA	8,3559	-0,1660	-0,6559

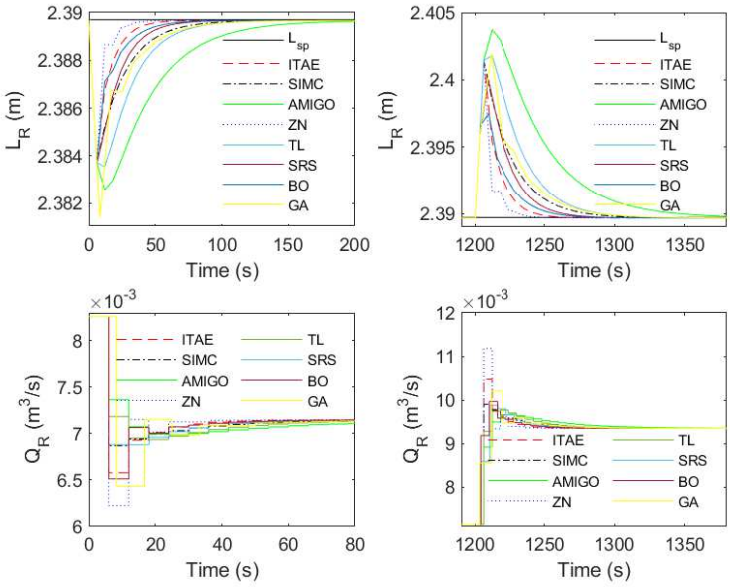


Figure 6.3: Level closed-loop response for different tuning methods

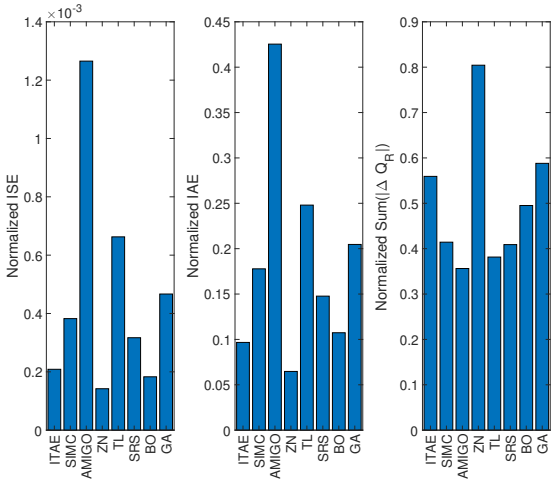


Figure 6.4: Level closed-loop performance metrics for different tuning methods

It can be seen that both standard model-based and model-free tuning methods show good disturbance rejection. Out of the first category, the ITAE tuning relation and the AMIGO method show the best and worst performance, respectively. This can be explained by the fact that the process model used in the tuning relations is perfect, because it was derived from the linear model (reactor mass balance equation) used to simulate the response. Because the AMIGO method explicitly includes robustness criteria (Åström and Hägglund (2006)), it considers the process model parameters as uncertain. On the other hand, the ITAE criterion tuning relation considers the process model to be accurately known, which is the case in this example. For model-free relations, Z-N show the lowest tracking errors of all the tuning methods. This comes at the expense of smooth control action, since the resulting controller is the most aggressive of all. T-L relations, which are more conservative, result in a compromise between performance and control action smoothness.

Regarding the proposed optimization-based tuning approach, it can be seen that of the three algorithms, GA shows the worst performance, both in terms of error and control action. The controller tuned by BO showed lower error, with slightly higher control action than MSRS. Both surrogate-based algorithms, especially BO, lead to controllers with performance comparable with ITAE and Z-N, however, with a less aggressive control action. This is the reflection of explicitly considering the trade-off in the optimization procedure.

6.3.3 Refinement of optimization-based tuning

After demonstrating the value of using SBO to tune the liquid level controller, the convergence properties of the two methods, BO and MSRS, are analysed. The maximum number of function evaluations is extended to 300 and the effect of initial sample size on convergence is determined. Two heuristics regarding the number of design variables, $10D$ (Loeppky et al. (2009)) and $5D$ (Liu et al. (2016)), are considered. In BO, the use of different acquisition functions is also compared.

It is important to state the goal of this analysis is not to thoroughly assess the convergence properties of these algorithms, but to refine the optimization procedure in order to find a better objective function value. A detailed analysis would require, for instance, a non-convex test problem with known optima and since the algorithms have stochastic properties, solving the optimization problem several times to obtain a distribution of objective function values. This analysis would require a great deal of time and since the goal was to refine the optimization procedure, a single solution was considered sufficient. Nonetheless, some remarks are able to be made from this analysis. The convergence of the algorithms is shown in Figure 6.5.

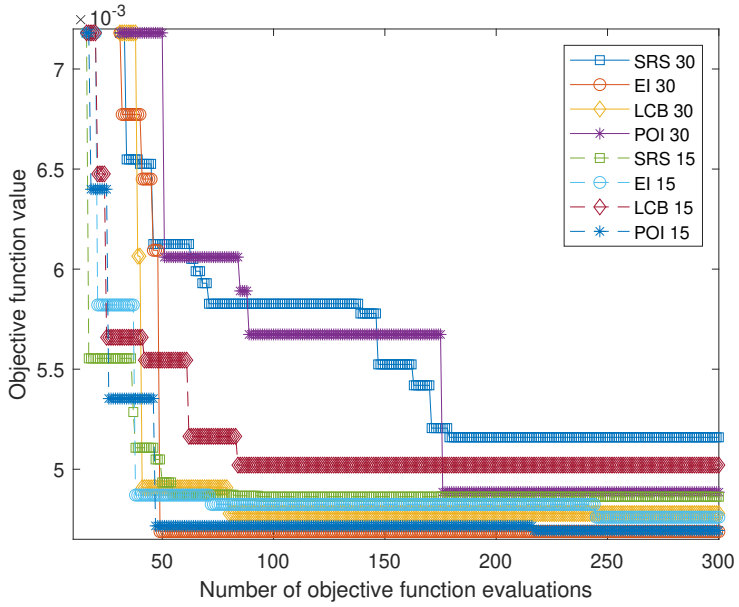


Figure 6.5: Convergence plot for the level controller parameter SBO

The use of an initial sample with a lower size generally leads to a lower objective function value within fewer function evaluations, with the exception of EI and LCB infill criteria. This could indicate that by initializing the adaptive sampling with a surrogate model with better approximation may condition the procedure to higher exploitation, by searching more locally near lower values, instead of exploring regions where better objective function values can be found. The effect of using a lower sample size seems to be less relevant when using EI instead of other criteria. This can perhaps be attributed to the fact that the use of this criterion automatically considers a trade-off between exploitation and explorations. As before, BO generally leads to finding a better objective function than MSRS, and the best controller parameters are considered to be the ones found with the EI criteria and 30 samples: Δt = 6,0209 s, K_P = -0,2197 and K_I = -0,0186. Therefore, the level controller with these parameters is used for the consequent closed loop simulations involved in the temperature controller tuning.

6.4 Temperature PID controller tuning

The temperature controller tuning serves as a more complex example. The system describing the reactor temperature is high order interactive, since the model has over 200 states and the reactor temperature affects the temperature of the cooling water and vice-versa. Because of multiple exothermic reactions, the temperature dynamic model is also highly non-linear. In addition, preliminary simulations showed that the reactor is open-loop unstable at the nominal operating point. This poses several difficulties, the most prominent being the derivation a linear transfer function process model to define the design space. Obtaining this model by linearisation proved difficult because the obtained linear model possessed a order of 208, with

numerator dynamics and a high number of unstable poles. Model-based order reduction techniques such as balanced truncation, were not useful. The chosen solution was to instead use system identification methods based on relay feedback.

When using relay feedback method for system identification, Liu et al. (2013) categorizes approaches into either Describing Function (DF) approximation, curve fitting or frequency response fitting. The latter method involves more extensive knowledge on control theory and requires more complex calculations (Liu et al. (2013)), so the choice was between the first two methods. To provide a simple and time efficient solution, the identification procedure must meet several requirements. The first is that it should, ideally, be simple to use and require only a single experimental test. The second is that no additional system knowledge, namely, process gain or delay, must be known in advance. Finally, the identification method should be applicable to unstable systems. Because of these requirements, several methods were ruled out. After searching different approaches, the Biased Relay method (Shen et al. (1996)) which meets all off these requirements, stood out. This method introduces an asymmetric oscillation by considering a relay with hysteresis. The asymmetry enables estimation of the process gain, which in other methods, is required to be known. The method only requires solving two simple equations to estimate the parameters of a First Order Plus Time Delay (FOPTD). These, along with the identification procedure results, are detailed in Appendix D. The identified parameters of the FOPTD model are $K = 2,5839 \times 10^5 K.s/m^3$, $\tau = 2,1377 \times 10^4 s$ and $\theta = 13{,}4$ s. As before, half the sampling time value is added to the First Order Plus Time Delay (FOPTD) process model:

$$\frac{T_R{}'(s)}{Q_{cin}{}'(s)} = \frac{K}{\tau s - 1} e^{-(\theta + \frac{\Delta t}{2})s} \tag{6.20}$$

According to Equation (6.4), the recommended sampling time range considering the identified process model is between 214 and 1070 seconds. Taking into account modelling errors and because these values seem very large and may lead to loss of information, or aliasing, these recommended values were not applied. Instead, Δt is assumed to be between 30 and 120 seconds. To provide a faster response, a PID controller is considered for the temperature control.

6.4.1 Standard model-based and model-free methods

Three model-based tuning relations were used, including the Rotstein-Lewin (RL) rules, based on a modified IMC method for unstable systems (Rotstein and Lewin (1991)), a criteria based on the minimization of the ISE criterion (Jhunjhunwala and Chidambaram (2001)) and a miscellaneous one (Sree and Chidambaram (2006)). These relations are presented in Table 6.5, considering:

$$\gamma = \lambda(\frac{\lambda}{\tau} + 2) \tag{6.21}$$

Table 6.5: Tuning relations for a PID controller and unstable FOPTD process model

Method	K_c	τ_I	τ_D
RL	$\frac{1}{K}\frac{\tau}{\lambda^2}(\gamma + 0,5\theta)$	$\gamma + 0,5\theta$	$\frac{0,5\gamma\theta}{\gamma+0,5\theta}$
ISE	$\frac{1}{K}(13 - 39,712\frac{\theta}{\tau})$	$0,856\tau e^{2,044\frac{\theta}{\tau}}$	$\tau(0,5643\frac{\theta}{\tau} + 0,0075)$
Sree	$\frac{1}{K}(4282(\frac{\theta}{\tau})^2 - 1334,6\frac{\theta}{\tau} + 101)$	$\tau(36,842(\frac{\theta}{\tau})^2 - 10,3\frac{\theta}{\tau} + 0,8288)$	$0,5\theta$

In the RL method, λ is a tunable parameter, related to the closed loop time constant. The value for this parameter can be chosen to guarantee closed-loop stability considering robustness to error in the process gain (Rotstein and Lewin (1991)). As recommended by Marchetti et al. (2001), since the real process gain is considered unknown, λ is determined graphically by choosing a value that maximizes worst-case relative gain uncertainty for a given value of θ/τ. Considering this ratio is approximately 0 for this process, λ was chosen as 0,01 τ.

After several trials using different sampling times for the relay tests, a sustained reactor temperature oscillation was not able to be achieved. This can be due to the fact that the sampler introduces an effective delay to the response and conditions regarding the value of θ/τ that are necessary to achieve sustained oscillation (Marchetti et al. (2001), Yu (2006)) are not met. However, because the true linear process model is not known, this assumption can not be confirmed. Nevertheless, this resulted in neglecting the use of model-free tuning methods.

Similarly to the level controller tuning, the effect of the sample time on controller performance was determined (Appendix E). As before, increasing the value of this parameter leads to worse controller performance. However, for this case, the difference between using a sample time of 30 or 60 seconds seems to be negligible for the RL method and the one by Sree and Chidambaram (2006). The controller tuned using the ISE relation resulted in poor performance. This may be due to the method not considering robustness directly in the optimization and the model used in this case is only a rough approximation. Because of the great difference in performance, this method is left out of the following analysis.

6.4.2 Comparison between standard tuning and optimization-based tuning

For the temperature controller, the optimization-based tuning approach remains the same as the one for the level controller, with the difference of using an initial design with 40 samples. Taking into consideration the results from the model-based methods, the design space is defined as:

$$30 \leq \Delta t(s) \leq 60 \tag{6.22}$$

$$-1 \times 10^{-3} \leq K_P \leq -1 \times 10^{-4} \tag{6.23}$$

$$-5 \times 10^{-6} \leq K_I \leq -5 \times 10^{-8} \tag{6.24}$$

$$-5 \times 10^{-2} \leq K_D \leq -1 \times 10^{-3} \tag{6.25}$$

Based on the initial sample in this design space, the median between the two objectives, f_1 and f_2, was approximately 0,0112. As before, w_2 is set equal to this value and w_1 is chosen to be 10.

Table 6.6: Temperature controller tuning optimization results for 100 function evaluations

Algorithm	Objective function value	Time (s)
GA	0,1937	328,9
SRS	0,1968	337,3
BO (EI)	0,1922	405,6

*Run on an Intel Core(TM) i7-8550U @1.8 GHz, 8GB RAM, Windows 10 64 bits

The results are similar to the ones for the level tuning, however, in this case GA was able to find a better value than MSRS, which is unexpected since SBO should be more efficient for the same number of objective function evaluations. However, due to the stochastic nature of the algorithm and since only one optimization run was performed, this result could be unreproducible. The controller parameters for the different tuning methods are presented in Table 6.7.

Table 6.7: Parameters for reactor temperature PID controller

Method	Δt (s)	K_P ($\times 10^4$)	K_I ($\times 10^6$)	K_D
RL ($\lambda = 0,01\tau$)	30,0000	-8,0360	-1,8104	-0,0110
Sree	60,0000	-3,7690	-0,0022	-0,0110
BO (EI)	30,0195	9,2362	-4,8362	-0,0369
SRS	57,1495	4,6603	-0,2258	-0,0331
GA	37,5325	9,1221	-4,1526	-0,0256

The temperature closed-loop response for the different controllers is presented in Figure 6.6 and the respective quantitative performance metrics in Figure 6.7. Considering the manipulated variable is the cooling water flowrate, a process utility with associated costs, the total use in cubic meters of cooling water during the simulation test is also quantified. This value is estimated considering the integral of Q_{cin} over the total simulation time, calculated using the trapezoidal rule.

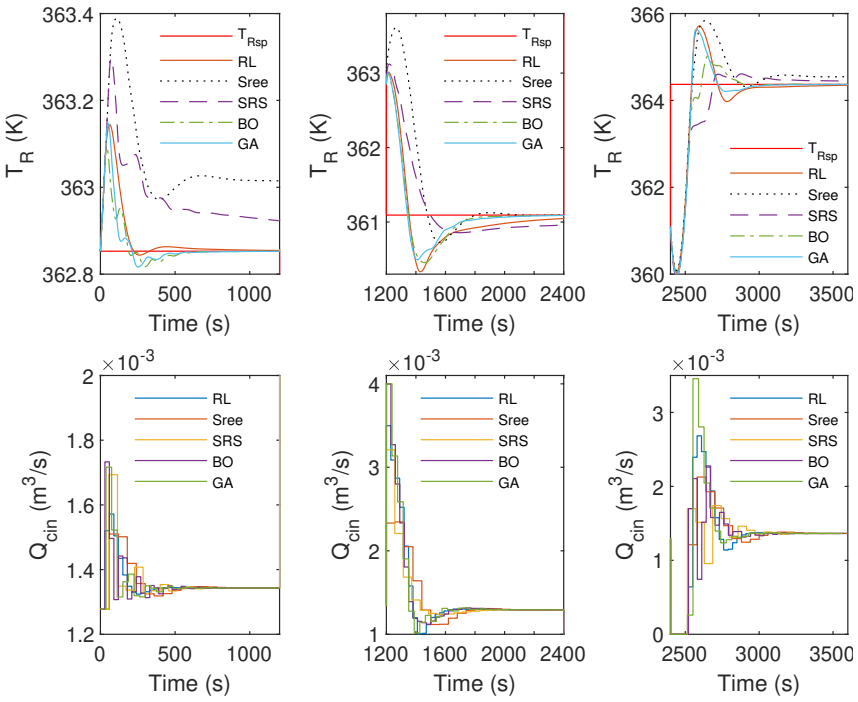


Figure 6.6: Temperature closed-loop response for different tuning methods

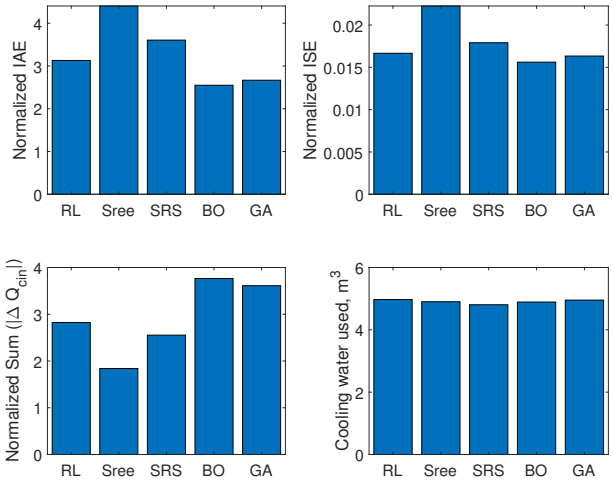


Figure 6.7: Temperature closed-loop performance metrics for different tuning methods

It can be seen that despite using a process model with low accuracy for the model-based tuning methods, both result in somewhat acceptable closed-loop performance, taking into account modelling errors. Of these two methods, the controller tuned by the RL rules shows superior performance, with faster settling times and lower overshoots, albeit with higher control action. For the optimization based methods, the controller tuned by the MSRS algorithm shows the worse performance, with greater error and settling times. Controllers tuned by BO and GA show similar values for ISE and IAE. Although the controller tuned by BO shows higher oscillation, and consequent increase in control action, it results in lower overshoots and a slight reduction in used cooling water. Regarding this last metric, one interesting observa-

tion is that despite differences in performance, the use of cooling water does not vary much for the different closed-loop simulations. This can be due to the temperature deviations from the different closed-loop simulations being relatively low coupled with the high heat removal capability provided by the cooling water lower temperature and high cooling coil heat transfer area. Overall, the controller tuned by BO shows the best compromise between performance and smooth control action.

6.4.3 Refinement of optimization-based tuning

Similarly to the level tuning procedure, the SBO approach is refined by increasing number of evaluations and using different initial sample sizes and acquisition functions.

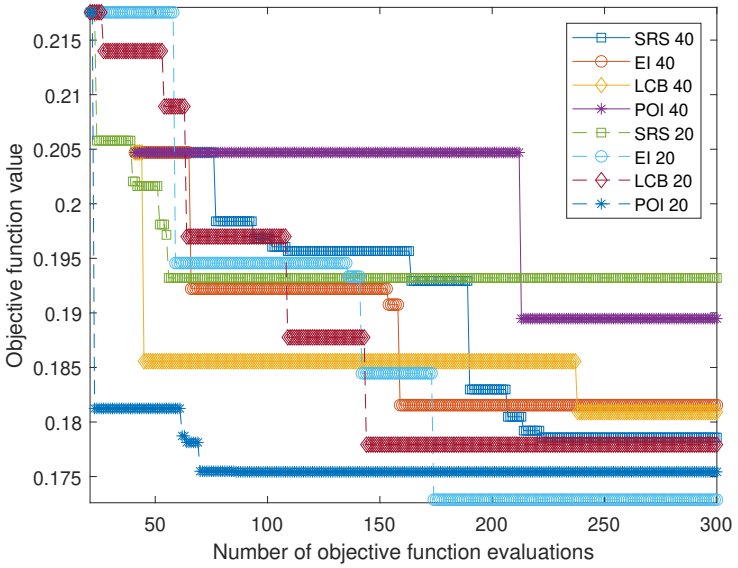


Figure 6.8: Convergence plot for the temperature controller parameter SBO

Contrary to previous results, MSRS displays better convergence and objective function values using a larger initial sample, while BO leads to better results using less samples. For BO, the POI criterion seems lead to higher exploitation, appearing stuck in local minima, spending many function evaluations with the same best objective function value. The same can also be said for the LCB criterion, when a larger initial sample is used. On the contrary, EI does not overly exploit. Similarly to the level results, best value was obtained for BO using the EI criterion, however, in this case, using fewer initial samples. The optimum controller parameters are $\Delta t = 30{,}0029$ s, $K_P = -9,0583 \times 10^{-4}$, $K_I = -2,5420 \times 10^{-7}$ and $K_D = -0,0214$.

6.5 Robustness analysis

Because no mathematical model can truly represent a physical process, robustness is an important issue in PID controller tuning and design. Despite not being considered explicitly in the proposed optimization-based tuning approach, it is still important to assess whether or not the tuned controller is robust, or in more simple terms, if its performances severely degrades due to changing operating conditions or uncertainty. Uncertainty is always present and can mainly arise from the model used in the controller tuning procedure or unaccounted process disturbances. Model uncertainty can be attributed to errors in the model parameters, unconsidered dynamic phenomena or when using a linear model, process non-linearity. Because the response of a higher fidelity non-linear model was used in the optimization-based tuning approach, this last source of model uncertainty is neglected. To assess the controller robustness, different scenarios are simulated:

- to reflect parametric uncertainty, the value of global heat transfer coefficient, U, varies 25% from its nominal value
- the temperature of both feed streams, T_A and T_B, are either at their lowest ($289K$) or highest ($300K$) daily values (Williams and Otto (1960)), reflecting changing operation conditions
- an unaccounted disturbance in which a positive or negative step change of $5K$ to the cooling water supply temperature, T_{Cin} occurs after 10 min of simulation

To determine whether the optimization-based tuning procedure reduced controller robustness, two sets of controllers are compared, one tuned using SBO (BO for both the level and temperature controller) and the other using standard tuning methods (ITAE rules for level controller and RL for temperature controller). The results are displayed on Figure 6.9. Because these changes do not affect the level dynamic behaviour, only the temperature closed-loop response is analysed.

The robustness of a PID controller can be quantified using several metrics based on linear control theory and system frequency response (Åström and Hägglund (2006)). However, since in this example the process model is non-linear simpler alternative metrics were chosen. The same performance metrics used before, ISE and sum of the rate of change in control action, are calculated for each scenario. Their values are presented in Figure 6.10. The assumption is that a more robust controller will display satisfactory performance even when uncertainty is present. In this sense, this approach can be though of as a worst-case uncertainty scenario type of analysis, as these were defined above. Alternatively, Monte Carlo simulations could also be used to assess the impact of uncertainty and quantify robustness in a probabilistic sense.

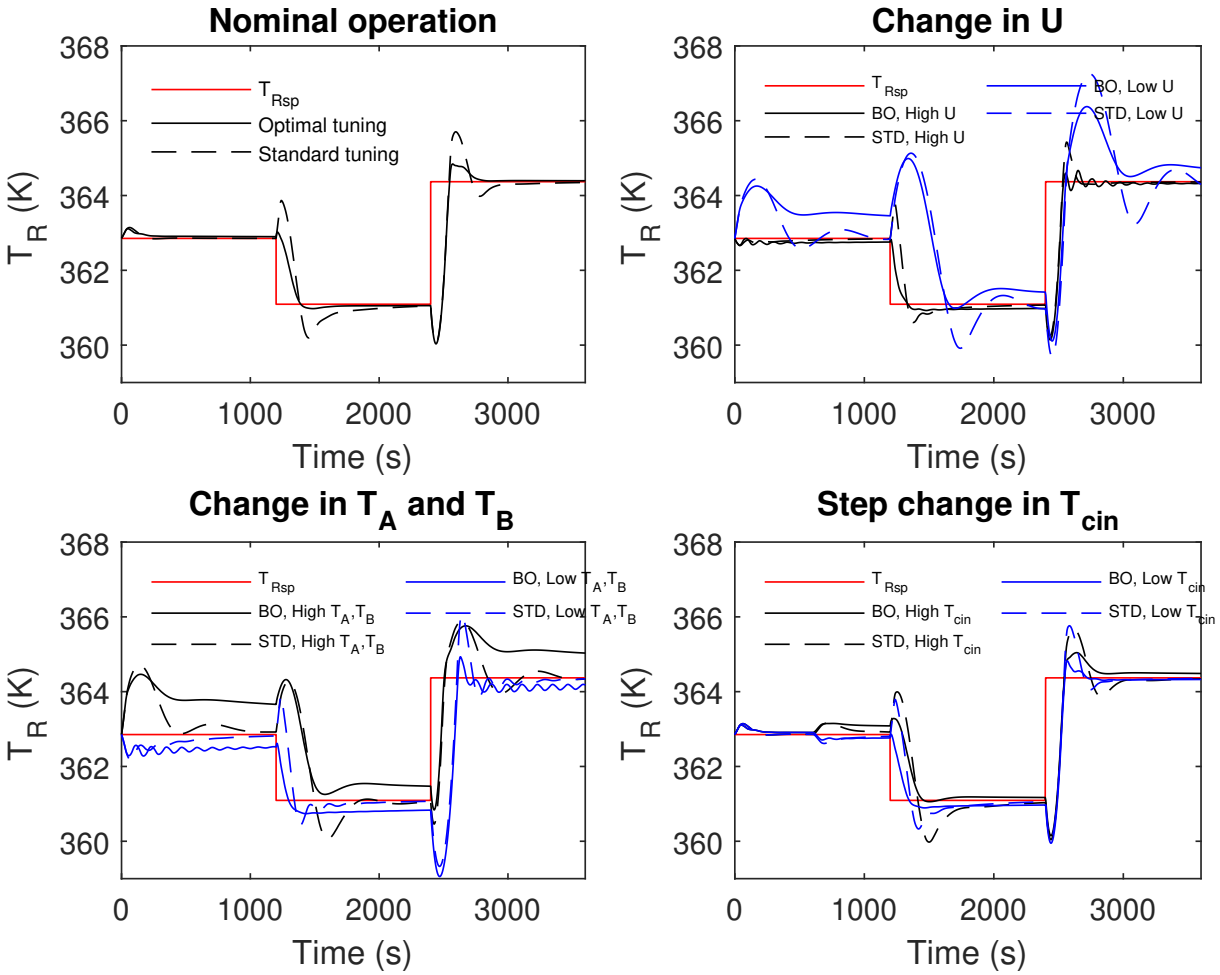


Figure 6.9: Closed-loop response for different scenarios (BO- controllers tuned with BO, STD - controllers tuned with standard methods)

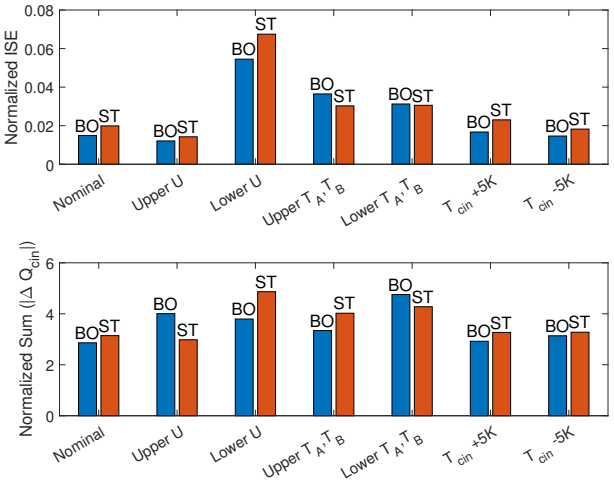


Figure 6.10: Increase in ISE from nominal operation(BO- controllers tuned by BO, ST - controllers tuned by standard tuning relations)

It can be seen that divergence from the nominal operation conditions results in worse controller performance, the only exceptions being the scenarios where U is lower and the cooling

water feed temperature is reduced. This is expected since both of these changes lead to a higher heat transfer rate and increased heat removal. Observing the closed-loop response for the different scenarios, a decrease in the value of U seems to have the greatest effect on the system response, followed by changes in the operating conditions. Comparatively, disturbances in the cooling water feed temperature and an increased U appear to not have as great impact.

Regarding parametric uncertainty and observing the value of ISE, the controller tuned by BO appears to be more robust than the one tuned by standard methods. For a higher value of U, however, this controller is more aggressive, resulting in increased temperature oscillations. This controller also shows superior ability to reject the cooling water temperature disturbance. However, the same does not happen when considering different operation conditions. In these scenarios, the controller tuned by BO shows worse performance than the one tuned with standard methods, with much higher deviations from the setpoint, overshoots, and for lower feed temperatures, increased oscillation.

In conclusion, in spite of displaying higher performance for some scenarios, this example serves to illustrate the need to incorporate robustness when tuning single-loop PID using optimization, taking into account, at least, changing operating conditions. An alternative would be to use more advanced single-loop control strategies based on adaptive and self-tuning controllers, where the controller parameters are estimated on-line based on a process model and acquired data (Seborg et al. (2017)). In this regard, off-line optimization-based tuning approaches using a simulated model, as the ones discussed so far, may not be needed.

Chapter 7

Conclusions and future work

7.1 Conclusions

First of all, it is important to mention the objective functions in the examples presented used in this work do not allow general and definitive conclusions regarding surrogate model performance, sample designs or sizes. Nonetheless, several remarks and recommendations about the use of Surrogate-Based Optimization (SBO) for the studied applications can be made.

The first case study considered the optimization of a cheap objective function with a known and convex form. After comparison with other derivative-based and Derivative Free Optimization (DFO) algorithms, SBO, both using static sampling and Bayesian Optimization (BO), found the optimum while the other two DFO algorithms failed to do so within a limited number of objective function evaluations, which displays higher efficiency of the surrogate-based methods. Although a derivative-based algorithm showed superior performance for a convex, smooth and cheap objective function, these conditions rarely apply to black-box problems. For this reason so there may be little use to apply derivative-based methods. It was shown that for a lower dimensional optimization problem, a better approach is to use surrogate models and adaptive sampling strategies with the goal of optimization. This avoids the problem of under or oversampling, and is a more time and effort efficient method. In addition, because the surrogate models are trained with static data, without accounting to changes in the black-box model parameters or input variable values, there is the need to refit a new surrogate model whenever such changes occur. Because of this, in addition to optimize the surrogate models, the whole model fitting procedure, including sample generation, model and hyper-parameter selection, needs to be performed once a process disturbance occurs. This can prove difficult or not worth the effort, especially whenever the optimization problem must be solved frequently or on-line, as in RTO. However, it must be emphasized that the use of SBO with adaptive sampling may not be as advantageous for larger, plant-wide RTO problems, with tens of decision variables, although that remains to be assessed in the future.

In the second case study, SBO was used to tune a PI and a PID controller using simulations from a complex non-linear dynamic model. The tuning problem was formulated as a multi-objective optimization problem. As it was expected, optimally tuned controllers displayed superior performance than those tuned by standard methods. However, this increase of performance did not came with an increase in effort due to the use of a systematic and almost automatic methodology. This methodology to determine the design space bounds and the objective function weights proved to be useful, simple and efficient, leading to a good controller performance after only one optimization run. Despite the need to perform at least one prior simulation, for system identification, after defining the design space, the tuning procedure is highly automatic. By considering the controller sampling time as a decision variable, all controller parameters are optimally selected, so no trial and error procedures are required to tune the controller, saving time and effort.

Based on the results of this work, the Biased-relay method is recommended for unstable system identification in order to obtain an approximate transfer function model, since it only required one simulation and was easy to use. In conjunction with model-based tuning methods that consider robustness, bounds on the decision variables were defined with no trial and error. By performing SBO within those bounds, good controller parameters were found. This way, it is recommended that when system identification is used to obtain a process model, model-based methods that take into account robustness, such as the AMIGO method for stable systems or the RL method for unstable systems, should be used.

After performing a robustness analysis consisting of a different scenarios with parametric uncertainty, unexpected disturbances and changing operating conditions, it was shown that the controllers with parameters obtained with BO showed higher robustness than the ones tuned by standard methods to model parametric uncertainty and unaccounted disturbances. However, performance was worse when considering changes in operating conditions. This illustrates the need to take into account uncertainty when performing the optimization-based tuning approach.

Overall, BO showed consistently superior results than MSRS. This comes at the cost of increased computational time, because of the auxiliary acquisition function optimization problem. However, if each objective function evaluation is time consuming, the increased computational time should not be a concern, since BO is more efficient in terms of evaluations. This algorithm showed the best results when using the EI criterion. This is because this acquisition function automatically considers the trade-off between exploration and exploitation. Regarding the effect of the initial sample size on SBO performance, there were mixed results. For BO using a lower sample size seemed to generally improve performance. However, if this sample is also used to obtain a notion of the scales between different objective, as was proposed, using a lower sample can bias the identification of these scales. In this regard, no definitive conclusion can be made.

7.2 Future Work

Despite demonstrating the advantages of using surrogate models to solve control-oriented optimization problems, the work of this thesis only touched a part of potential applications. Additional interesting case studies and approaches are identified for future work.

7.2.1 Approximation of the Pareto set

In this work, the multi-objective optimization problem was solved by defining each objective importance a priori, through the use of the weighted-sum method. A alternative to better reflect the relative importance of each objective is to choose the solution based on a approximation of the Pareto set. To this end, and considering SBO, several multi-objective BO algorithms can be used (Rojas-Gonzalez and Van Nieuwenhuyse (2019)).

7.2.2 Optimization-based tuning of more complex control structures

Beyond tuning single-loop PID controllers, SBO could also be used to tune controllers in more complex structure, for instance cascade or multi-loop PID controllers. Also, it would be interesting to use this optimization approach to tune linear or non-linear MPC. Despite the existence of some work in these regards (cf. chapter 4), demonstrating the advantages of this procedure on case studies considering chemical processes, on which references are few and scarce, should be interesting. If there is a lower layer of PID control under the MPC, the benefits of a simultaneous tuning procedure for both controllers based on optimization could also be tested. These examples would involve higher dimensional optimization problems, a factor which would have to be taken into consideration. Another problem may be the increase in the number of optimization objectives and the definition of the trade-off between them.

7.2.3 Surrogate-based optimization for NMPC

As mentioned in chapter 4, an interesting and relatively unused idea is to use SBO to solve the optimization problem in NMPC. Since the problem must be solved on-line, in addition to efficiency in the number of objective function evaluations, the algorithm overhead time, due to additional calculations, would have to also be taken into consideration. This would mean that BO, which showed better results but with higher computational time, would perhaps result in lower performance than other more fast surrogate-based algorithms, for instance MSRS or trust-region algorithms, such as the ones mentioned in Chapter 4.3. The problem would also require dealing with constraints, possibly non-linear.In addition, the number of decision variables is proportional with the number of manipulated variables and the length of the control horizon. An increase in either or both of these inevitably escalates the dimensionality of the op-

timization problem, which would have to be taken into account when selecting surrogate-based approaches.

Surrogate-based DFO algorithms could be compared with those traditionally used, relying on derivative information to determine if the surrogate approach possesses merit and if the assumption that the objective function evaluation is the limiting factor is valid. The approach could also be tested for situations where a prediction model is embedded in an external simulation software, such as Aspen Plus®.

7.2.4 Deal with uncertainty in optimization

Last, and certainly not the least, it would be important and interesting to include a measure of uncertainty when solving the controller tuning optimization problem using surrogate models. This would be important not only for single loop, but multiple loop PID controllers as well.

How to include uncertainty and design the controller explicitly considering robustness is an interesting and challenging problem. There are already optimization-based tuning approaches which optimize the controller parameters considering a measure of robustness (Åström and Hägglund (2006)). However, this measure is based on notions from linear control theory and frequency response, which would require modifications or different approaches when considering a non-linear process model. The use of high-fidelity or black-box type simulations for the controller tuning procedure would also have to be considered when devising solutions. Based on a recent review by Hüllen et al. (2020), ideas from robust optimization, stochastic programming or discrepancy modelling used in a SBO framework could be investigated. Solutions found in recent works regarding the use of BO(cf. Hüllen et al. (2020) for references therein) under uncertainty can also be investigated.

Bibliography

Alhajeri, M. and Soroush, M. (2020). Tuning Guidelines for Model-Predictive Control. *Ind. Eng. Chem. Res.*, 59(10):4177–4191.

Åström, K. J. and Hägglund, T. (1984). Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, 20(5):645–651.

Åström, K. J. and Hägglund, T. (2006). *Advanced PID Control*. ISA, Research Triangle Park, NC.

Åström, K. J. and Wittenmark, B. (2011). *Computer Controlled Systems: Theory and Design*. Prentice Hall, Upper Saddle River, NJ, 3rd edition.

Bansal, S., Calandra, R., Xiao, T., Levine, S., and Tomlin, C. J. (2017). Goal-driven dynamics learning via Bayesian optimization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5168–5173, Melbourne, VIC. IEEE.

Bemporad, A. (2020). Global optimization via inverse distance weighting and radial basis functions. *arXiv:1906.06498 [cs, math, stat]*.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E. N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20.

Berkenkamp, F., Krause, A., and Schoellig, A. P. (2020). Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics. *arXiv:1602.04450 [cs]*.

Berner, J., Hägglund, T., and Åström, K. J. (2016). Asymmetric relay autotuning – Practical features for industrial use. *Control Engineering Practice*, 54:231–245.

Berner, J., Soltesz, K., Hägglund, T., and Åström, K. J. (2018). An experimental comparison of PID autotuners. *Control Engineering Practice*, 73:124–133.

Bhosekar, A. and Ierapetritou, M. (2018). Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, 108:250–267.

Biegler, L. T., Lang, Y.-d., and Lin, W. (2014). Multi-scale optimization for process systems engineering. *Computers & Chemical Engineering*, 60:17–30.

Bikmukhametov, T. and Jäschke, J. (2020). Combining machine learning and process engineering physics towards enhanced accuracy and explainability of data-driven models. *Computers & Chemical Engineering*, 138:106834.

Boukouvala, F., Misener, R., and Floudas, C. A. (2016). Global optimization advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO. *European Journal of Operational Research*, 252(3):701–727.

Box, E. P. and Draper, N. R. (1987). *Empirical Model Building and Response Surfaces*. Wiley, New York.

Box, G. E. and Behnken, D. (1960). Some new three level designs for the study ofquantitative variables. *Technometrics*, 2(4):455–475.

Brochu, E., Cora, V. M., and de Freitas, N. (2010). A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *arXiv:1012.2599 [cs]*.

Caballero, J. A. and Grossmann, I. E. (2008). An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE J.*, 54(10):2633–2650.

Chakrabarty, A., Dinh, V., Corless, M. J., Rundell, A. E., Zak, S. H., and Buzzard, G. T. (2017). Support Vector Machine Informed Explicit Nonlinear Model Predictive Control Using Low-Discrepancy Sequences. *IEEE Trans. Automat. Contr.*, 62(1):135–148.

Chugh, T., Sindhya, K., Hakanen, J., and Miettinen, K. (2017). A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Comput*, 23(9):3137–3166.

Cox, D. D. and John, S. (1997). SDO: A statistical method for global optimization. In *Multidisciplinary Design Optimization: State of the Art*, pages 315–329.

Cozad, A., Sahinidis, N. V., and Miller, D. C. (2014). Learning surrogate models for simulation-based optimization. *AIChE J.*, 60(6):2211–2227.

Crombecq, K., De Tommasi, L., Gorissen, D., and Dhaene, T. (2009). A novel sequential design strategy for global surrogate modeling. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 731–742, Austin, TX, USA. IEEE.

Crombecq, K., Laermans, E., and Dhaene, T. (2011). Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. *European Journal of Operational Research*, 214(3):683–696.

Dæhlen, J. S., Eikrem, G. O., and Johansen, T. A. (2014). Nonlinear model predictive control using trust-region derivative-free optimization. *Journal of Process Control*, 24(7):1106–1120.

Dias, L. S. and Ierapetritou, M. G. (2020). Integration of planning, scheduling and control problems using data-driven feasibility analysis and surrogate models. *Computers & Chemical Engineering*, 134:106714.

Díaz-Manríquez, A., Toscano, G., Barron-Zambrano, J. H., and Tello-Leal, E. (2016). A Review of Surrogate Assisted Multiobjective Evolutionary Algorithms. *Computational Intelligence and Neuroscience*, 2016:1–14.

Eason, J. P. and Biegler, L. T. (2016). A trust region filter method for glass box/black box optimization. *AIChE J.*, 62(9):3124–3136.

Fang, H. and Horstemeyer, M. F. (2006). Global response approximation with radial basis functions. *Engineering Optimization*, 38(4):407–424.

Fang, K. T., Lin, D. K., Winker, P., and Zhang, Y. (2000). Uniform design: Theory andapplication. *Technometrics*, 42(3):237–248.

Feng, Z., Shen, W., Rangaiah, G. P., and Dong, L. (2018). Proportional-Integral Control and Model Predictive Control of Extractive Dividing-Wall Column Based on Temperature Differences. *Ind. Eng. Chem. Res.*, 57(31):10572–10590.

Fisher, R. A. (1926). The arrangement of field experiments. *J. Ministry Agric. Great Britain*, 33(4):503–513.

Forbes, J. F. (1994). *Model Structure and Adjustable Parameter Selection for Operations Optimization*. PhD Thesis, McMaster University, Ontario, Canada.

Forbes, M. G., Patwardhan, R. S., Hamadah, H., and Gopaluni, R. B. (2015). Model Predictive Control in Industry: Challenges and Opportunities. *IFAC-PapersOnLine*, 48(8):531–538.

Forgione, M., Piga, D., and Bemporad, A. (2019). Efficient Calibration of Embedded MPC. *arXiv:1911.13021 [cs, eess, math]*, 86(3).

Forrester, A. I. and Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1-3):50–79.

Forrester, A. I. J., Sóbester, A., and Keane, A. J. (2008). *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, first edition.

Frazier, P. I. (2018). A Tutorial on Bayesian Optimization. *arXiv:1807.02811 [cs, math, stat]*.

Garud, S. S., Karimi, I. A., Brownbridge, G. P., and Kraft, M. (2018). Evaluating smart sampling for constructing multidimensional surrogate models. *Computers & Chemical Engineering*, 108:276–288.

Garud, S. S., Karimi, I. A., and Kraft, M. (2017a). Design of computer experiments: A review. *Computers & Chemical Engineering*, 106:71–95.

Garud, S. S., Karimi, I. A., and Kraft, M. (2017b). Smart Sampling Algorithm for Surrogate Model Development. *Computers & Chemical Engineering*, 96(4):103–114.

Gelbart, M. A., Snoek, J., and Adams, R. P. (2014). Bayesian Optimization with Unknown Constraints. *arXiv:1403.5607 [cs, stat]*.

Greenhill, S., Rana, S., Gupta, S., Vellanki, P., and Venkatesh, S. (2020). Bayesian Optimization for Adaptive Experimental Design: A Review. *IEEE Access*, 8:13937–13948.

Gugercin, S. and Antoulas, A. C. (2004). A Survey of Model Reduction by Balanced Truncation and Some New Results. *International Journal of Control*, 77(8):748–766.

Gutmann, H. M. (2001). A Radial Basis Function Method for Global Optimization. *Journal of Global Optimization*, 19(3):201–227.

Halton, J. H. (1964). Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, 7(12):701–702.

Hammersley, J. M. (1960). Monte Carlo methods for solving multivariable problems. *Ann. N. Y. Acad. Sci.*, 86(3):844–874.

Haykin, S. S. (2009). *Neural Networks and Learning Machines*. Prentice Hall, New York, 3rd edition.

Hennig, P. and Schuler, C. J. (2012). Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837.

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. (2014). Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, pages 918–926.

Himmelblau, D. M. (2008). Accounts of Experiences in the Application of Artificial Neural Networks in Chemical Engineering. *Ind. Eng. Chem. Res.*, 47(16):5782–5796.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, USA.

Hüllen, G., Zhai, J., Kim, S. H., Sinha, A., Realff, M. J., and Boukouvala, F. (2020). Managing uncertainty in data-driven simulation-based optimization. *Computers & Chemical Engineering*, 136:106519.

Jhunjhunwala, M. and Chidambaram, M. (2001). PID controller tuning for unstable systems by optimization method. *Chemical Engineering Communications*, 185:91–113.

Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *J. Stat. Plan. Inference*, 26(2):131–148.

Jones, D. R. (2001). A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4):345–383.

Jones, D. R., Perttunen, C. D., and Stuckman, B. E. (1993). Lipschitzian optimization without the Lipschitz constant. *J Optim Theory Appl*, 79(1):157–181.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492.

Kadupitiya, J., Sun, F., Fox, G., and Jadhao, V. (2020). Machine learning surrogates for molecular dynamics simulations of soft materials. *Journal of Computational Science*, 42:101107.

Kajero, O. T., Chen, T., Yao, Y., Chuang, Y.-C., and Wong, D. S. H. (2017). Meta-modelling in chemical process system engineering. *Journal of the Taiwan Institute of Chemical Engineers*, 73:135–145.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1945, Perth, Australia.

Keßler, T., Kunde, C., McBride, K., Mertens, N., Michaels, D., Sundmacher, K., and Kienle, A. (2019). Global optimization of distillation columns using explicit and implicit surrogate models. *Chemical Engineering Science*, 197:235–245.

Khosravi, M., Behrunani, V., Smith, R. S., Rupenyan, A., and Lygeros, J. (2020). Cascade Control: Data-Driven Tuning Approach Based on Bayesian Optimization. *arXiv:2005.03970 [cs, eess]*.

Khosravi, M., Eichler, A., Schmid, N., Smith, R. S., and Heer, P. (2019). Controller Tuning by Bayesian Optimization An Application to a Heat Pump. In *2019 18th European Control Conference (ECC)*, pages 1467–1472, Naples, Italy. IEEE.

Kim, S. H. and Boukouvala, F. (2020). Machine learning-based surrogate modeling for data-driven optimization: A comparison of subset selection for regression techniques. *Optim Lett*, 14(4):989–1010.

Kong, J., Eason, J. P., Chen, X., and Biegler, L. T. (2020). Operational Optimization of Polymerization Reactors with Computational Fluid Dynamics and Embedded Molecular Weight Distribution Using the Iterative Surrogate Model Method. *Ind. Eng. Chem. Res.*, 59(19):9165–9179.

Kummer, A., Nagy, L., and Varga, T. (2020). NMPC-based control scheme for a semi-batch reactor under parameter uncertainty. *Computers & Chemical Engineering*, 141(4).

Laskawski, M. and Wcislik, M. (2016). Sampling Rate Impact on the Tuning of PID Controller Parameters. *International Journal of Electronics and Telecommunications*, 62(1):43–48.

Latha, K., Rajinikanth, V., and Surekha, P. M. (2013). PSO-Based PID Controller Design for a Class of Stable and Unstable Systems. *ISRN Artificial Intelligence*, 2013:1–11.

Liu, H., Ong, Y.-S., and Cai, J. (2018). A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. *Struct Multidisc Optim*, 57(1):393–416.

Liu, H., Xu, S., Ma, Y., Chen, X., and Wang, X. (2016). An adaptive Bayesian sequential sampling approach for global metamodeling. *J. Mech. Des.*, 138(1).

Liu, T., Wang, Q.-G., and Huang, H.-P. (2013). A tutorial review on process identification from step or relay feedback test. *Journal of Process Control*, 23(10):1597–1623.

Loeppky, J. L., Sacks, J., and Welch, W. J. (2009). Choosing the Sample Size of a Computer Experiment: A Practical Guide. *Technometrics*, 51(4):366–376.

Lovelett, R. J., Dietrich, F., Lee, S., and Kevrekidis, I. G. (2019). Some manifold learning considerations toward explicit model predictive control. *AIChE J*, 66(5).

Luyben, W. L. (2007). *Chemical Reactor Design and Control*. AIChE ; Wiley-Interscience, [New York] : Hoboken, N.J.

Marchetti, G., Scali, C., and Lewin, D. R. (2001). Identification and control of open-loop unstable processes by relay methods. *Automatica*, 37:2049–2055.

Marco, A., Hennig, P., Bohg, J., Schaal, S., and Trimpe, S. (2016). Automatic LQR tuning based on Gaussian process global optimization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 270–277, Stockholm. IEEE.

Marler, R. and Arora, J. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395.

Marler, R. T. and Arora, J. S. (2010). The weighted sum method for multi-objective optimization: New insights. *Struct Multidisc Optim*, 41(6):853–862.

Marzat, J. and Piet-Lahanier, H. (2012). Design of nonlinear MPC by Kriging-based optimization. *IFAC Proceedings Volumes*, 45(16):1490–1495.

McBride, K. and Sundmacher, K. (2019). Overview of Surrogate Modeling in Chemical Process Engineering. *Chemie Ingenieur Technik*, 91(3):228–239.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

Metropolis, N. and Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341.

Morari, M. and Lee, J. H. (1999). Model predictive control: Past, present and future. *Computers and Chemical Engineering*, page 16.

Mukhopadhyay, T., Mahata, A., Dey, S., and Adhikari, S. (2016). Probabilistic Analysis and Design of HCP Nanowires: An Efficient Surrogate Based Molecular Dynamics Simulation Approach. *Journal of Materials Science & Technology*, 32(12):1345–1351.

Navia, D., Gutiérrez, G., and de Prada, C. (2013). Nested Modifier-Adaptation for RTO in the Otto Williams Reactor. *IFAC Proceedings Volumes*, 46(32):123–128.

Negrellos-Ortiz, I., Flores-Tlacuahuac, A., and Gutiérrez-Limón, M. A. (2018). Dynamic optimization of a cryogenic air separation unit using a derivative-free optimization approach. *Computers & Chemical Engineering*, 109:1–8.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *Comput. J.*, 7:308–313.

Nentwich, C. and Engell, S. (2019). Surrogate modeling of phase equilibrium calculations using adaptive sampling. *Computers & Chemical Engineering*, 126:204–217.

Nery Júnior, G. A., Martins, M. A., and Kalid, R. (2014). A PSO-based optimal tuning strategy for constrained multivariable predictive controllers with model uncertainty. *ISA Transactions*, 53(2):560–567.

O'Dwyer, A. (2009). *Handbook of PI and PID Controller Tuning Rules*. Imperial College Press ; Distributed by World Scientific Pub, London : Hackensack, NJ, 3rd ed edition.

Park, S., Na, J., Kim, M., and Lee, J. M. (2018). Multi-objective Bayesian optimization of chemical reactor design using computational fluid dynamics. *Computers & Chemical Engineering*, 119:25–37.

Piga, D., Forgione, M., Formentin, S., and Bemporad, A. (2019). Performance-oriented model learning for data-driven MPC design. *IEEE Control Syst. Lett.*, 3(3):577–582.

Plackett, R. L. and Burman, J. P. (1946). The design of optimum multifactorial experiments. *Biometrika*, 33(4):305–325.

Poulin, É. and Pomerleau, A. (1996). PID tuning for integrating and unstable processes. *IEE Proceedings – Control Theory and Applications*, 143:429–435.

Powell, M. J. D. (1994). A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In *Gomez S., Hennart JP. (Eds) Advances in Optimization and Numerical Analysis*, volume 275 of *Mathematics and Its Applications*, pages 51–67. Springer, Dordrecht.

Powell, M. J. D. (2009). The BOBYQA algorithm for bound constrained optimization without derivatives. http://www.damtp.cam.ac.uk/user/na/NA_papers/NA2009_06.pdf.

Rafiei, M. and Ricardez-Sandoval, L. A. (2020). Integration of design and control for industrial-scale applications under uncertainty: A trust region approach. *Computers & Chemical Engineering*, 141:107006.

Ramasamy, V., Sidharthan, R. K., Kannan, R., and Muralidharan, G. (2019). Optimal Tuning of Model Predictive Controller Weights Using Genetic Algorithm with Interactive Decision Tree for Industrial Cement Kiln Process. *Processes*, 7(12):938.

Rangaiah, G. P., Feng, Z., and Hoadley, A. F. (2020). Multi-Objective Optimization Applications in Chemical Process Engineering: Tutorial and Review. *Processes*, 8(5):508.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge.

Rawlings, J. B., Mayne, D. Q., and Diehl, M. M. (2017). *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, Madison, Wisconsin, 2nd edition edition.

Regis, R. G. (2016). Trust regions in Kriging-based optimization with expected improvement. *Engineering Optimization*, 48(6):1037–1059.

Regis, R. G. and Shoemaker, C. A. (2007). A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions. *INFORMS Journal on Computing*, 19(4):497–509.

Regis, R. G. and Shoemaker, C. A. (2013). Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5):529–555.

Regis, R. G. and Wild, S. M. (2017). CONORBIT: Constrained optimization by radial basis function interpolation in trust regions. *Optimization Methods and Software*, 32(3):552–580.

Rios, L. M. and Sahinidis, N. V. (2012). Derivative-free optimization: A review of algorithms and comparison of software implementations. *J Glob Optim*, 56(3):1247–1293.

Rojas-Gonzalez, S. and Van Nieuwenhuyse, I. (2019). A survey on kriging-based infill algorithms for multiobjective simulation optimization. *Computers & Operations Research*, 116:104869.

Rosenblatt, F. (1957). The Perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory.

Rotstein, G. E. and Lewin, D. R. (1991). Simple PI and PID tuning for open-loop unstable systems. *Industrial & Engineering Chemistry Research*, 30:1864–1869.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and Analysis of Computer Experiments. *Statist. Sci.*, 4(4):409–423.

Schillinger, M., Hartmann, B., Skalecki, P., Meister, M., Nguyen-Tuong, D., and Nelles, O. (2017). Safe Active Learning and Safe Bayesian Optimization for Tuning a PI-Controller. *IFAC-PapersOnLine*, 50(1):5967–5972.

Schreiter, M., Nguyen-Tuong, D., Eberts, M., Bischoff, B., Market, H., and Toussaint, M. (2015). Safe exploration for active learning with gaussian processes. In *Machine Learning and Knowledge Discovery in Databases*, pages 133–149.

Seborg, D. E., Edgar, T. F., Mellichamp, D. A., and Doyle III, F. J. (2017). *Process Dynamics and Control*. John Wiley & Sons, Inc.,, Hoboken, NJ, fourth edition.

Shen, S.-H., Wu, J.-S., and Yu, C.-C. (1996). Use of biased-relay feedback for system identification. *AIChE J.*, 42(4):1174–1180.

Shokry, A., Dombayci, C., and Espuña, A. (2016). Multiparametric Metamodels for Model Predictive Control of Chemical Processes. In *Computer Aided Chemical Engineering*, volume 38, pages 937–942. Elsevier.

Singhal, M., Marchetti, A. G., Faulwasser, T., and Bonvin, D. (2016). Real-Time Optimization Based on Adaptation of Surrogate Models. *IFAC-PapersOnLine*, 49(7):412–417.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control*, 13:291–309.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

Sobol, I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i MatematicheskoiFiziki*, 7(4):784–802.

Sree, R. P. and Chidambaram, M. (2006). *Control of Unstable Systems*. lpha Science International Ltd., Oxford.

Straus, J. (2018). *Optimal Operation of Integrated Chemical Processes With Application to the Ammonia Synthesis*. PhD thesis, Norwegian University of Science and Technology, Norway.

Straus, J. and Skogestad, S. (2018). Surrogate model generation using self-optimizing variables. *Computers & Chemical Engineering*, 119:143–151.

Straus, J. and Skogestad, S. (2019). A new termination criterion for sampling for surrogate model generation using partial least squares regression. *Computers & Chemical Engineering*, 121(2):75–85.

Sui, Y., Gotovos, A., Burdick, J. W., and Krause, A. (2015). Safe exploration for optimization with gaussian processes. In *Proceedigns of the 32nd International Conference on Machine Learning*, Lille, France.

Tyreus, B. D. and Luyben, W. L. (1992). Tuning PI controllers for integrator/dead time processes. *Ind. Eng. Chem. Res.*, 31(11):2625–2628.

Visioli, A. (2001). Optimal tuning of PID controllers for integral and unstable processes. *IEE Proceedings - Control Theory and Applications*, 148(2):180–184.

Wang, G. G. and Shan, S. (2007). Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design*, 129(4):370–380.

Wang, Y. and Shoemaker, C. A. (2014). A General Stochastic Algorithmic Framework for Minimizing Expensive Black Box Objective Functions Based on Surrogate Models and Sensitivity Analysis. *arXiv:1410.6271 [stat]*.

Williams, T. J. and Otto, R. E. (1960). A Generalized Chemical Processing Model for the Investigation of Computer Control. page 16.

Wilson, Z. T. and Sahinidis, N. V. (2017). The ALAMO approach to machine learning. *Computers & Chemical Engineering*, 106:785–795.

Yang, K., Gaida, D., Back, T., and Emmerich, M. (2015). Expected hypervolume improvement algorithm for PID controller tuning and the multiobjective dynamical control of a biogas plant. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 1934–1942, Sendai, Japan. IEEE.

Yang, S., Kiang, S., Farzan, P., and Ierapetritou, M. (2018). Optimization of Reaction Selectivity Using CFD-Based Compartmental Modeling and Surrogate-Based Optimization. *Processes*, 7(1):9.

Yu, C.-C. (2006). *Autotuning of PID Controllers*. Springer-Verlag London Limited, Germany, second edition.

Zhai, J. and Boukouvala, F. (2019). Nonlinear variable selection algorithms for surrogate modeling. *AIChE J*, 65(8).

Zhu, M., Bemporad, A., and Piga, D. (2020). Preference-based MPC calibration. *arXiv:2003.11294 [math]*.

Ziegler, J. G. and Nichols, N. B. (1942). Optimum Settings for Automatic Controllers. *Trans. ASME,*, 64(759).

Appendices

A Williams-Otto reactor dynamic model

A simplified PI&D of the reactor considered in this work is shown below.

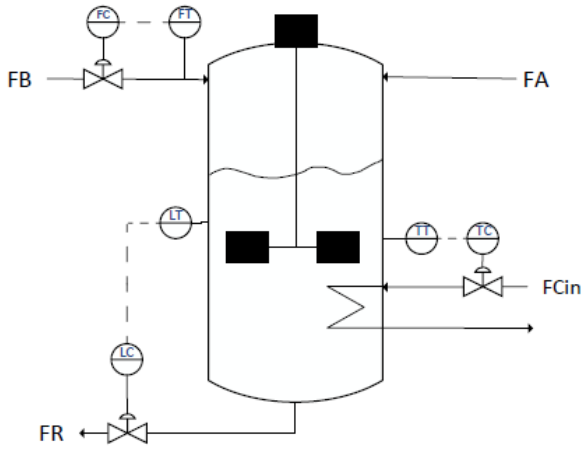


Figure A.1: Simplified PI&D of reactor

Modifications to the original model in Williams and Otto (1960) included calculating the specific heat capacity of the reactor mixture, C_{pR} (Kummer et al. (2020)), neglecting both the internal heating coils and the recycle stream, and the addition of equations to describe the reactor level and cooling water temperature dynamics. These equations were derived using mass and energy balances, with the following considerations:

- The physical properties of both the reactor mixture and cooling water are constant and do not present temperature dependence
- The reactor is considered a perfect cylinder with constant area
- Inside the coil flows pure water
- The coil wall is made of metal with a high thermal conductivity and a low thickness, so resistance to heat transfer can be considered low and the wall temperature dynamics can be neglected
- Due to the low thickness of the coil wall, its outer diameter is assumed to be equal to its inner diameter
- The water flowing inside the coil is considered an incompressible fluid
- The internal coil is permanently full of water, so mass dynamics are neglected
- The cooling water flow pattern is considered plug flow, where the contribution of heat dispersion phenomena, both in the axial and radial directions, is considered negligible when compared to convection

A. Williams-Otto reactor dynamic model

The model of the system is described by the following set of differential equations:

$$\rho_R \pi \frac{{D_R}^2}{4} \frac{dL_R}{dt} = F_A + F_B - \rho_R Q_R \tag{1}$$

$$m_R \frac{dX_A}{dt} = F_A - \rho_R Q_R X_A - k_1 X_A X_B m_R \tag{2}$$

$$m_R \frac{dX_B}{dt} = F_B - \rho_R Q_R X_B - k_1 X_A X_B m_R - k2 X_B X_C m_R \tag{3}$$

$$m_R \frac{dX_C}{dt} = -\rho_R Q_R X_C + 2k_1 X_A X_B m_R - 2k_2 X_B X_C m_R - k_3 X_C X_P m_R \tag{4}$$

$$m_R \frac{dX_E}{dt} = -\rho_R Q_R X_E + 2k_2 X_B X_C m_R \tag{5}$$

$$m_R \frac{dX_G}{dt} = -\rho_R Q_R X_G + 1,5 k_3 X_C X_P m_R \tag{6}$$

$$m_R \frac{dX_P}{dt} = -\rho_R Q_R X_P + k_2 X_B X_C m_R - 0,5 k_3 X_C X_P m_R \tag{7}$$

$$m_R C_{pR} \frac{dT_R}{dt} = F_A C_{pA}(T_A - T_R) + F_B C_{pB}(T_B - T_R) + \tag{8}$$

$$2k_1 X_A X_B m_R(-\Delta H_{R1}) + 3k_2 X_B X_C m_R(-\Delta H_{R2}) + 1,5 k_3 X_C X_P m_R(-\Delta H_{R3}) + Q_{cool}$$

$$\rho_W C_{pW} \frac{dT_C(z,t)}{dt} = -\frac{4Q_{cin}}{\pi {D_C}^2} \rho_W C p_W \frac{\partial T_C}{\partial z} + \frac{4U}{D_C}(T_R - T_C) \tag{9}$$

With initial conditions:

$$L_R(0) = L_R^0 \tag{10}$$

$$X_i(0) = X_i^0, \qquad i = A, B, C, E, G, P \tag{11}$$

$$T_R(0) = T_R^0 \tag{12}$$

And boundary conditions:

$$Z = 0: \qquad T_C(0,t) = T_{Cin} \tag{13}$$

$$Z = L_C: \qquad \frac{\partial T_C(L_C,t)}{\partial z} = 0 \tag{14}$$

The model is composed of a equation system with 8 ODEs and 1 PDE. To enable a simultaneous solution, the last equation was solved using the method of lines by discretization of the partial derivative in Equation (9) along the spatial coordinate, which is approximated using backwards finite differences:

$$\frac{\partial T_C(z)}{\partial z} \approx \frac{T_{Ck} - T_{Ck-1}}{\Delta z} \tag{15}$$

The reaction rates, k_i, are described by Arrhenius law:

$$k_i = A_i \exp\left(\frac{-Ea_i}{T_R}\right), i = 1, 2, 3 \tag{16}$$

The value of the reactor diameter, D_R, is not provided by Williams and Otto (1960). Considering

a recommended aspect ratio of $L_R = 2D_R$ (Luyben (2007)), the diameter can be estimated by:

$$V_T = \frac{\pi}{4}{D_R}^2 L_R \Leftrightarrow V_T = \frac{\pi}{2}{D_R}^3 \Leftrightarrow D_R = \sqrt[3]{\frac{2V_T}{\pi}} \tag{17}$$

The total volume, V_T , corresponds to the sum of reactor mixture volume, V_R, and the internal coil volume, V_{coil}:

$$V_T = V_R + V_{coil} \Leftrightarrow V_T = \frac{m_R}{\rho_R} + \frac{\pi}{4}{D_C}^2 L_C \tag{18}$$

The mass hold up, m_R was considered the same as the one in Chapter 4, 2105,2 kg. The heat capacity of the reactor mixture, C_{pR} is a function of each component mass fraction, X_i and individual heat capacity, Cp_i:

$$Cp_R = \sum_{i=1}^{6} Cp_i X_i \tag{19}$$

The total heat exchanged between the cooling water and the reactor mixture, Q_{cool}, can be determined through the integral of heat transfer along the coil length, L_C, considering a constant heat transfer coefficient, U, and coil diameter, D_C:

$$Q_{cool} = U\pi D_C \int_0^{L_C} (T_C(z,t) - T_R(t))dz \tag{20}$$

The integral in the equation above was approximated using numerical integration and Riemann's sum:

$$Q_{cool} \approx U\pi D_C \sum_{k=1}^{N} (T_{Ck}(t) - T_R(t))\Delta z \tag{21}$$

Where N is the number of discrete lumps and T_{Ck} the temperature of the coil lump k. The model parameters are presented in the table below.

Table A.1: Williams-Otto reactor dynamic model parameters (Williams and Otto (1960), Forbes (1994)), Kummer et al. (2020)

Parameter	Value	Parameter	Value
$A_1(s^{-1}kgA)$	$1,6599 \times 10^6$	C_{pA} $(Jkg^{-1}K^{-1})$	2262
A_2 $(s^{-1}kgB)$	$7,2117 \times 10^8$	C_{pB} $(Jkg^{-1}K^{-1})$	2119
A_3 $(s^{-1}kgC)$	$2,6745 \times 10^{12}$	C_{pC} $(Jkg^{-1}K^{-1})$	1744
Ea_1 (K)	6666,7	C_{pE} $(Jkg^{-1}K^{-1})$	2052
Ea_2 (K)	8333,3	C_{pG} $(Jkg^{-1}K^{-1})$	2357
Ea_3 (K)	11111	C_{pP} $(Jkg^{-1}K^{-1})$	2204
ρ_R (kgm^{-3})	800,92	C_{pW} $(Jkg^{-1}K^{-1})$	4186
ΔH_{R1} $(kJkg^{-1})$	-263,8	ρ_W (kgm^{-3})	1000
ΔH_{R2} $(kJkg^{-1})$	-158,3	U $(Wm^{-2}K^{-1})$	800
ΔH_{R3} $(kJkg^{-1})$	-226,3	D_C (m)	0,0254
L_C (m)	116,4336	D_R (m)	1,1960

Since the reactor temperature and component mass fractions values at the nominal point are known, as they were determined by steady-state optimization, the only variable values that need to be calculated are the level, cooling water exit temperature and flowrate. The level value at steady state is arbitrary, since it does not depend on other variables. Therefore it was determined considering the reactor mass hold-up, 2105,2 kg. The cooling water exit temperature, T_{Cout}, and flowrate, Q_{cin}, can be determined by a set of sequential calculations (Luyben (2007)). First the heat that must be removed, Q_{cool} to maintain a constant T_R, is determined by solving Equation (8) in steady-state. At steady-state, Q_{cool} can also be described by the logarithmic mean:

$$Q_{cool} = \frac{(T_R - T_{Cin}) - (T_R - T_{Cout})}{ln\left(\frac{T_R - T_{Cin}}{T_R - T_{Cout}}\right)} \tag{22}$$

Solving Equation (22) in order to T_{Cout} gives $T_{Cout} = 344,4K$. Q_{cin}, can be determined through a steady-state energy balance:

$$Q_{cin} = \frac{Q_{cool}}{\rho_W Cp_W (T_{Cout} - T_{Cin})} \tag{23}$$

Because the cooling water temperature varies along the length of the coil, the spatial temperature profile at steady state must be known. This profile can be determined solving equation Equation (9) at steady-state, which is an ODE that has an analytical solution:

$$\begin{aligned} 0 &= -\frac{4Q_{Cin}}{\pi {D_C}^2}\rho_W Cp_W \frac{\partial T_C(z)}{\partial z} + \frac{4U}{D_C}(T_R - T_C(z)) \Leftrightarrow \\ Tc(z) &= T_R - (T_R - T_{Cin}) \exp\left(\frac{-\pi U D_C Z}{\rho_W Cp_W Q_{cin}}\right) \end{aligned} \tag{24}$$

Z depends on the number of discrete steps along the axial coordinate. To determine this number, a sensitivity analysis was taken in which the relative absolute error between the steady state value of Q_{cool} (from Equation (8)) and the approximation of Equation (21), as a function of the number of steps, using Equation (24) to determine the temperature spatial profile.

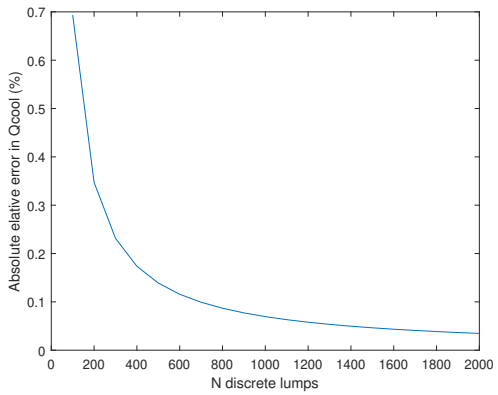


Figure A.2: Relative error as a function of the number of discretization steps

Based on this analysis, 200 discretization steps were chosen, which resulted in a absolute relative error of approximately $0,35\%$ from the rigorous Q_{cool} value of $297,6kW$. The value of the input variables for the nominal scenario are given on the table below.

Table A.2: Nominal values for input variables

Input variable	**Value**	**Input variable**	**Value**	**Input variable**	**Value**
$T_A(K)$	294,3	$\bar{Q_R}(L/s)$	8,2593	$\bar{Q_C}(L/s)$	1,2778
$T_B(K)$	294,3	$Q_{Rmax}\ (L/s)$	12	$Q_{cinmax}\ (L/s)$	4
$T_{cin}(K)$	288,75	$Q_{Rmin}\ (L/s)$	0	$Q_{cinmin}(L/s)$	0

It must be noted that the upper limits on the flowrates Q_R and Q_{cin} were considered as a base example.

B Surrogate model selection and validation for chapter 5

B.1 Polynomial regression

Polynomial regression models were implemented using *fit*. In these models, the first independent variable is the reactant B flowrate, F_B and the second is the reactor temperature T_R. Firstly, several polynomial model types were tested, as to asses the most adequate. Since the initial sample size is 20, and the number of samples must be greater than the number of coefficients to estimate, several types of polynomials are ruled out. Preliminary tests also showed that first or second order terms lead to higher prediction error. The remaining options are compared in figure below.

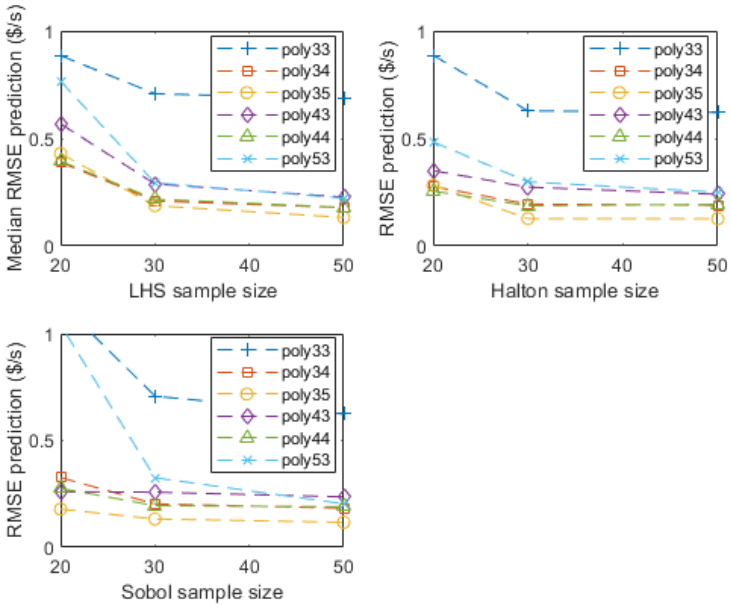


Figure B.3: Comparison between different polynomial models for different sampling plans

Increasing the number of samples does not appear to have a significant effect on lowering the prediction error, with only slight differences in prediction errors using 30 or 50 training samples. It can be seen that a polynomial model with third order terms for F_B and fifth order terms for T_R outperforms other models across all designs, reason for choosing this model instead of all the others.

B.2 Kriging

The kriging models were implemented using *fitrgp*, MLE was performed using the default MATLAB® optimizer.

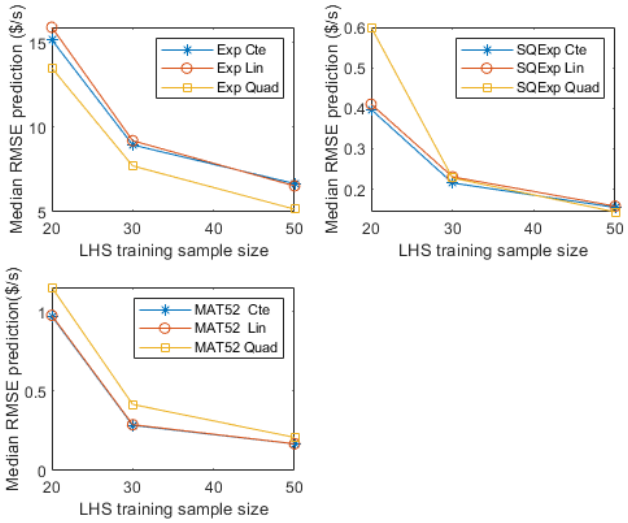


Figure B.4: Comparison of RMSE using different kriging kernels and trends using LHS

The exponential kernel is much less accurate than the other two kernels, by at least an order of magnitude, and the squared exponential kernel appears to have higher accuracy than the Matérn 5/2. Other apparent conclusion is that the performance using a linear trend is similar when using a constant or a quadratic trend, except for the exponential kernel. This would indicate that, in specific this case, increasing model complexity, by using a polynomial trend, does not lead to significant improvements in model accuracy. To assess these conclusions, the error using squared exponential and Matérn 5/2 kernels is compared for the other deterministic designs, that do not present randomness.

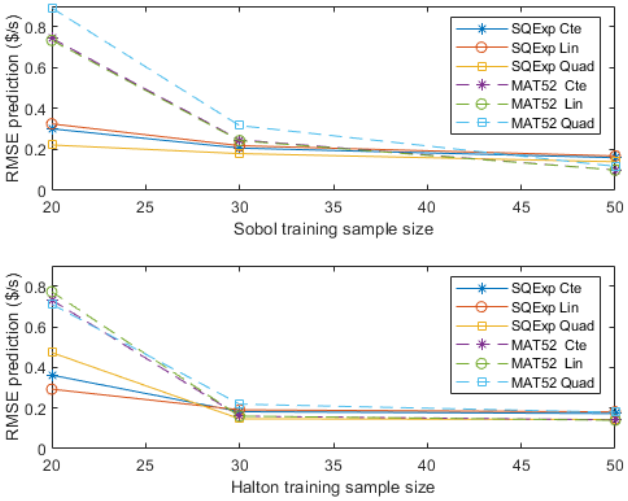


Figure B.5: Comparison of Squared Exponential and Matérn 5/2 kernel using Sobol and Halton designs

Observing the results of both designs, it can be seen the prediction error when using a Matérn kernel is two times larger for the lowest sample size and only shows better performance

for an increase in number of points, for both designs. This leads to favour the use of a squared exponential kernel for a smaller sample and the Matérn 5/2 kernel for a larger sample. As before, the performance when using a constant or linear trend for the squared exponential kernel is comparable, however, in this case, a quadratic trend improves the prediction accuracy. These results are similar to the ones reported by Bhosekar and Ierapetritou (2018) for several test problems. Since the increase in accuracy for the quadratic trend comes at the expense of a more complex model, which requires more training points to estimate additional model hyper-parameters of the quadratic trend, and increases computational time, the use of a quadratic trend instead of a constant trend seems to not be worth the trade-off. For the Matérn kernel, the use of a polynomial trend does not appear to lower prediction error.

B.3 Artificial Neural Networks

Feedforward ANN were implemented using the *feedforwardnet*, with the weights and biases estimated considering the mean squared error as the objective loss function and the back-propagation optimization problem was solved using the fast Levenberg-Marquardt algorithm, by default in MATLAB®. The number of training iterations was left at default, 1000. The weights were initialized always using the same random number generator, for reproducibility. To assess the impact of network architecture on prediction error, various types of networks were compared for Sobol designs with 20, 30 and 50 samples. One and two hidden layers with 10 nodes in each layer were considered, with two of the most used activation functions, the hyperbolic tangent and the logistic function. For simplicity sake, in the case of two hidden layers, the number of neurons and activation functions are identical in each layer.

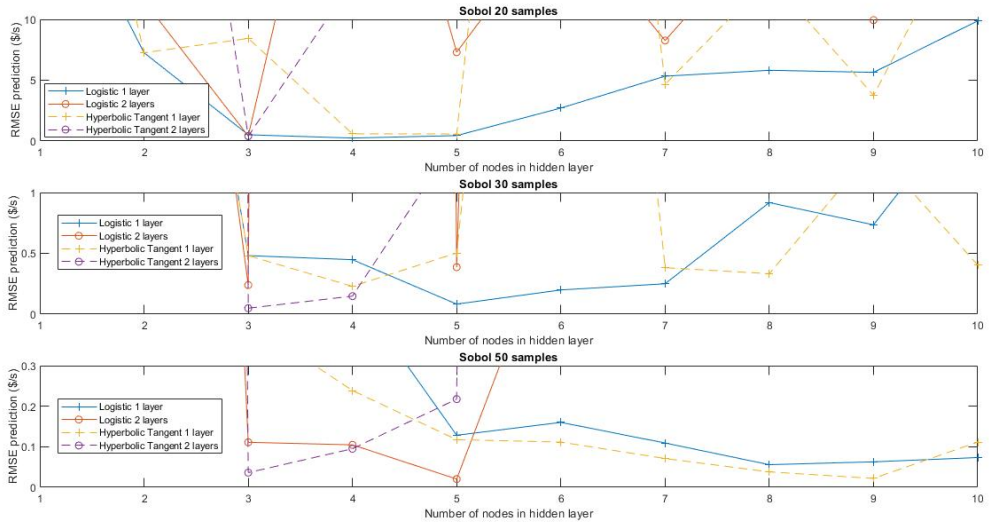


Figure B.6: Effect of neural network architecture and sample size on prediction error.

When a lower sample size is used, the networks with a lower number of nodes (three to five)

present a lower prediction error. Also networks with only one hidden layer generally perform better than those with two. This is because bigger networks have a larger number of weights and bias to estimate and therefore require a bigger amount of training data to prevent overfitting. As the initial sample size increases, this fact becomes apparent, with bigger networks showing a lower prediction error. Looking at the two activation functions, it appears that for a single hidden layer, better performance is achieved with the use of a logistic function for lower sample size and hyperbolic tangent for a higher sample size. However, the impact of different activation functions when two hidden layers are used is not as obvious.

All things considered, the results above show that for the largest number of samples, the network architecture which presents the lowest error is a network with two hidden layers with 5 nodes each, using the logistic activation function, with a RMSE of $0,02034\$/s$. However, a closer look reveals that a one hidden layer with 9 nodes using the hyperbolic tangent shows the second best performance, with a RMSE of $0,02196\$/s$. Comparing the number of parameters in each network, the first one has a total of 51 parameters (10 and 30 weights for the input and hidden layer, respectively, and 11 biases) while the second network has 37 parameters (18 and 9 weights for the input and hidden layer, respectively, and 10 biases). Considering that a model with fewer coefficients is generally more favourable, and that difference in performance is marginal between the simpler network with only one hidden layer is chosen.

B.4 Radial basis functions

RBF models were implemented using *newrbe*, which chooses all data points as centres for the basis functions and estimates the weights solving the linear equation system. A bias on the network output is also estimated. Only the Gaussian radial basis function (see Equation (2.24)) was considered for this case. The value of the width, γ, constant for all neurons, was adjusted by testing several values for different sample designs.

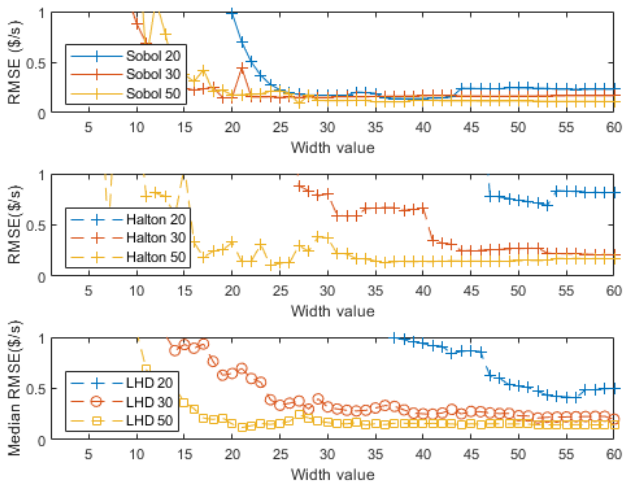


Figure B.7: Effect of width in Gaussian RBF prediction error for different sampling plans

It seems that the value of the width, γ has a greater impact on prediction error when the initial design has fewer samples and increasing this value improves accuracy, up to a point. As the number of samples increase, increasing the width appears to not have as significant effect on prediction error, eventually stagnating. This is expected since the the width value is related to the vector distance for which each Gaussian neuron responds in the input space, so that with fewer samples, each neuron has to respond to a greater input space area so that new inputs are detected. With a lower number of samples, the area does not need to be as large. Overall, the best performance is achieved with a γ value of 27 for a Sobol design, leading to the lowest prediction error across the three initial sample sizes.

B.5 Support Vector Regression

SVR models were fitted through *fitrsvm* and the quadratic optimization problem is solved using the default Sequential Minimal Optimization algorithm. Taking this into account, the use of a Gaussian kernel, where γ is set to 1 by default in *fitrsvm*, and polynomial kernels of second to fifth order were tested for different initial designs and sample sizes.

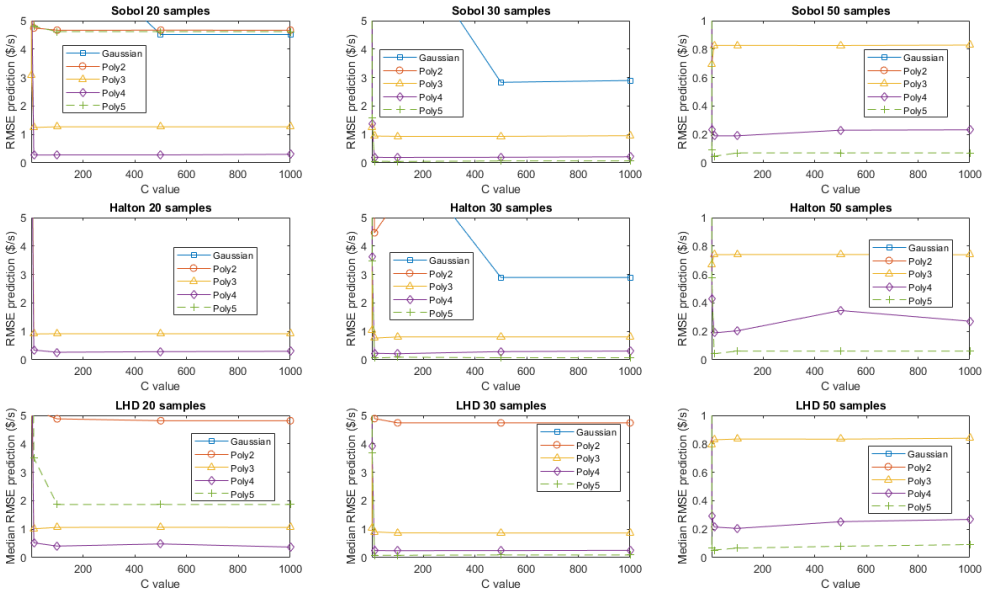


Figure B.8: Effect of C and different kernels on SVR prediction error for different sampling plans

It can be seen that the impact of increasing the parameter C is only significant for the Gaussian kernel, and in the case of polynomial kernels, increasing C above 100 does not lead to a decrease in prediction error. This indicates that when using a polynomial kernel, increasing model complexity with a higher value of C leads to worse performance, increasing fitting time with the same, or higher, prediction error. Also, for these kernels, as the sample size increases,

the error is lower when a lower value of C is used, meaning that for a larger sample, the use of a more complex model is not favourable.

Regarding different kernels, it can be seen that the Gaussian kernel shows worse performance compared to polynomial kernels above second order. Within the different polynomial kernels, the fourth order kernel is more accurate for a lower sample size but as the initial sample size increases, the fifth order kernel leads to a much lower prediction error. Considering these results, the use of a fifth order polynomial with a C value of 10 is favoured above all others, leading the lowest error for higher sample sizes.

C Simulink flowsheets used in closed-loop simulations

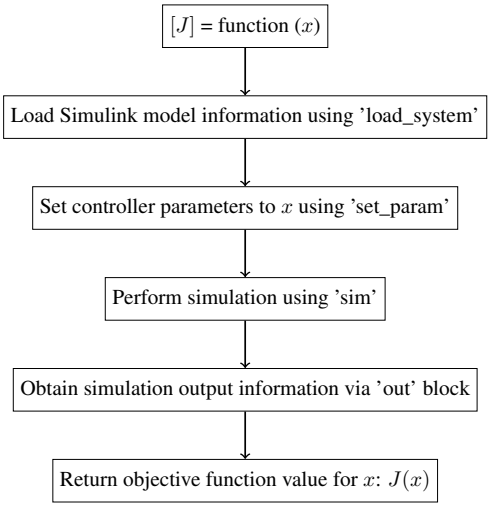


Figure C.9: Representation of MATLAB® function used in simulation optimization with Simulink®

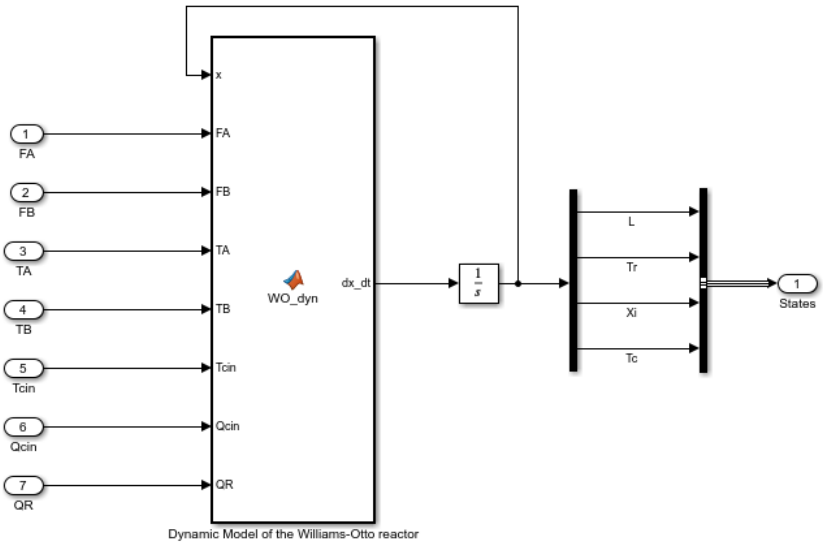


Figure C.10: Simulink model of the Williams-Otto reactor

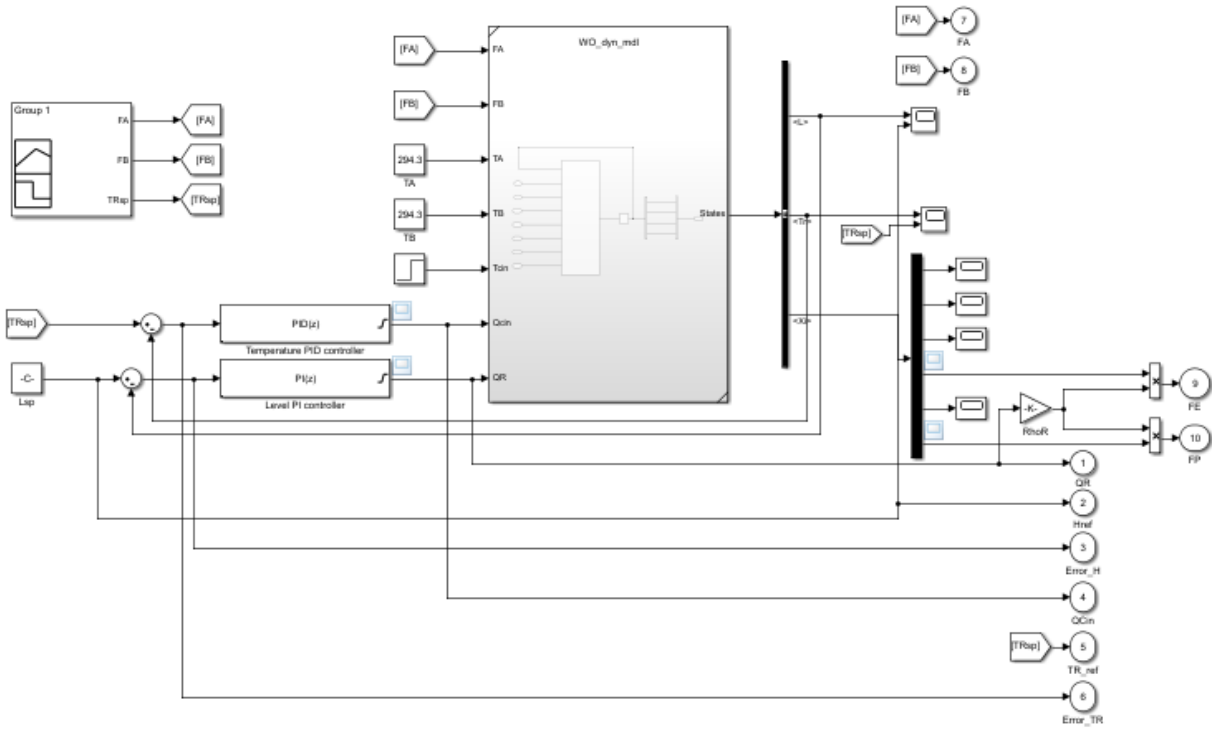


Figure C.11: Simulink flowsheet for PID control

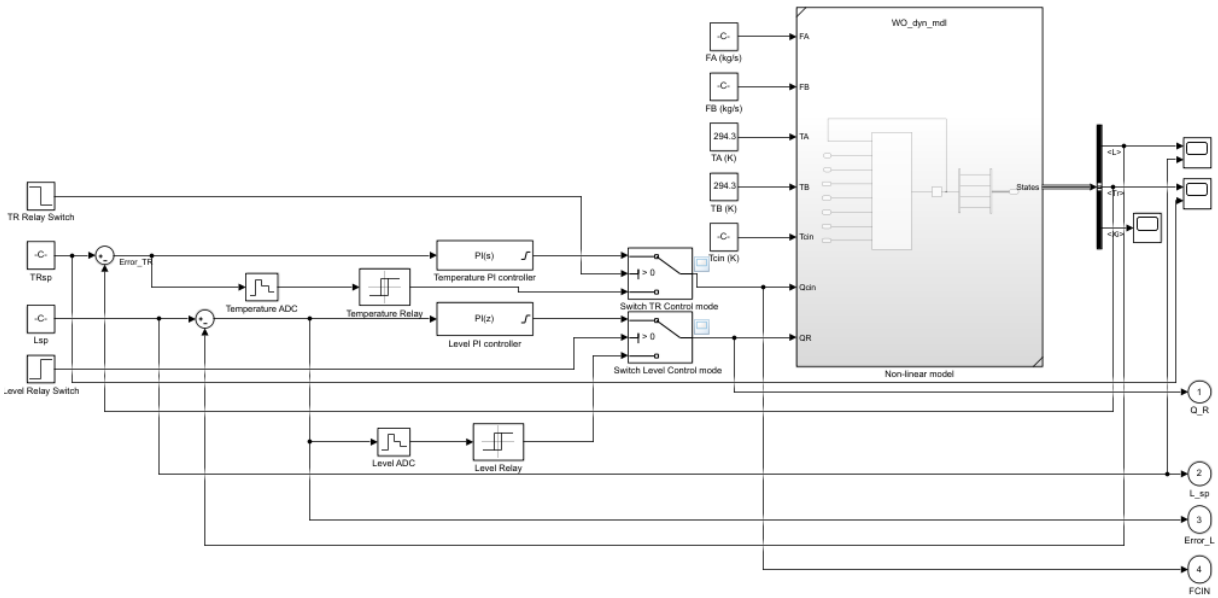


Figure C.12: Simulink flowsheet for Relay tests

D Biased-relay method description and results

In this method, a relay with hysteresis in one direction is used:

$$u(t) = \begin{cases} d & , \text{ if } e(t) \geq T \\ -d & , \text{ if } e(t) < 0 \end{cases} \tag{25}$$

The addition of a hysteresis generates an asymmetric oscillation, which allows approximating the process gain as:

$$K = -\frac{\int_0^P y(t)dt}{\int_0^P u(t)dt} \tag{26}$$

The integral are evaluated over one period of oscillation, P. After calculating the process gain (Equation (26)), the time constant, τ and delay, θ can be estimated, after modification for unstable system identification (Marchetti et al. (2001)) by:

$$\tau = \frac{\sqrt{\left(\frac{K}{K_{cu}}\right)^2 - 1}}{\omega_u} \tag{27}$$

$$\theta = \frac{arctg(\omega_u \tau)}{\omega_u} \tag{28}$$

Based on the system response, K_{cu} is estimated using Equation (6.11) and the ultimate frequency,ω_u is:

$$\omega_u = \frac{2\pi}{P_u} \tag{29}$$

Because the relay is not ideal and the controlled variable does not display symmetric oscillation, both parameters present increased error in the approximation. Considering a test with T as 0,1 K and d as $1,3430 \times 10^{-3} m^3/s$, the system response is:

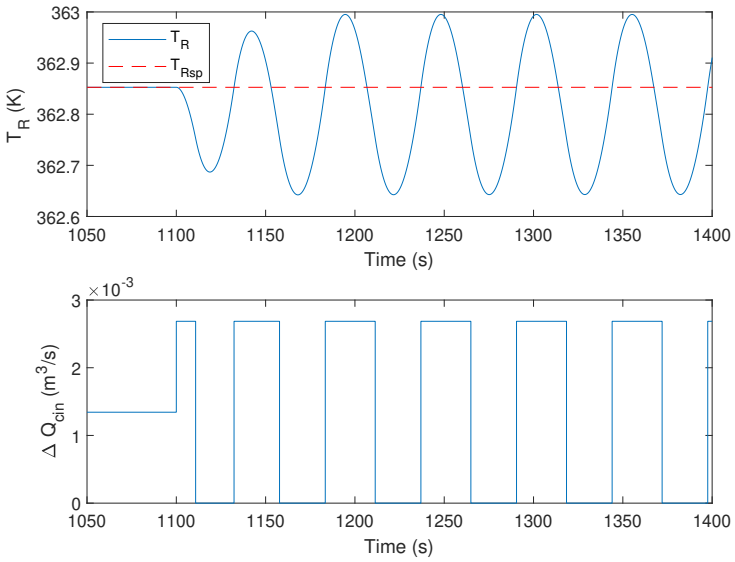


Figure D.13: Biased Relay test

Based on this response, K_{cu} is 0,0097 and P_u is 53,53 seconds. A comparison between the closed-loop response to a unitary step setpoint change of the identified model with the original non-linear model is made in the figure below:

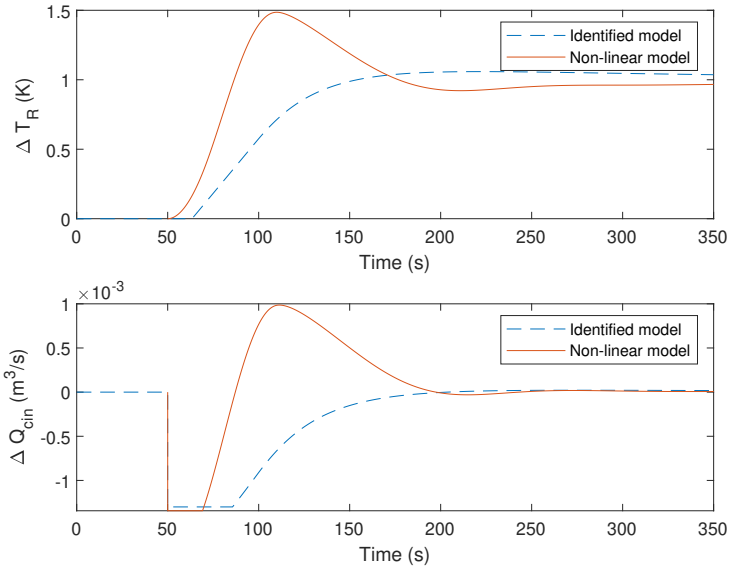


Figure D.14: Identified and original model closed-loop response to a step change in setpoint

E Additional closed-loop simulation figures

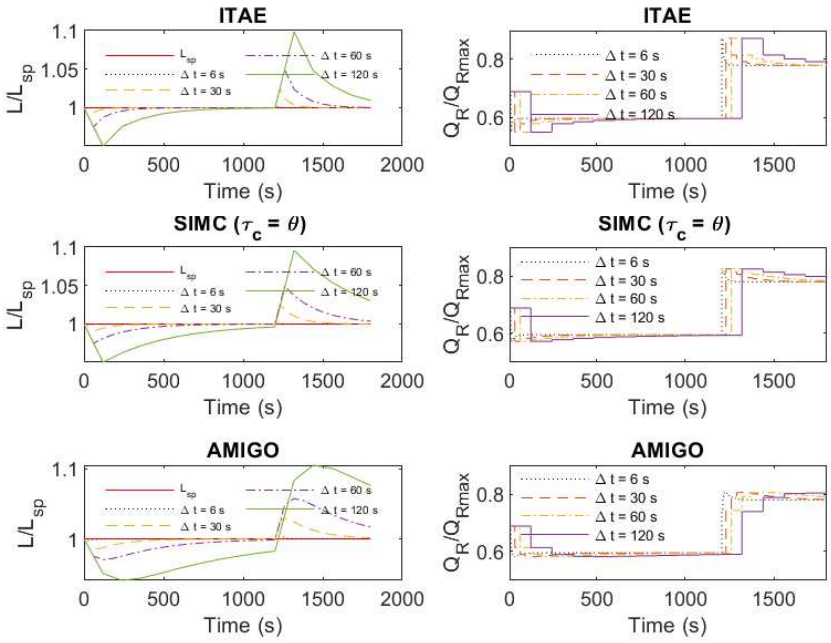


Figure E.15: Level closed-loop response for different sampling times

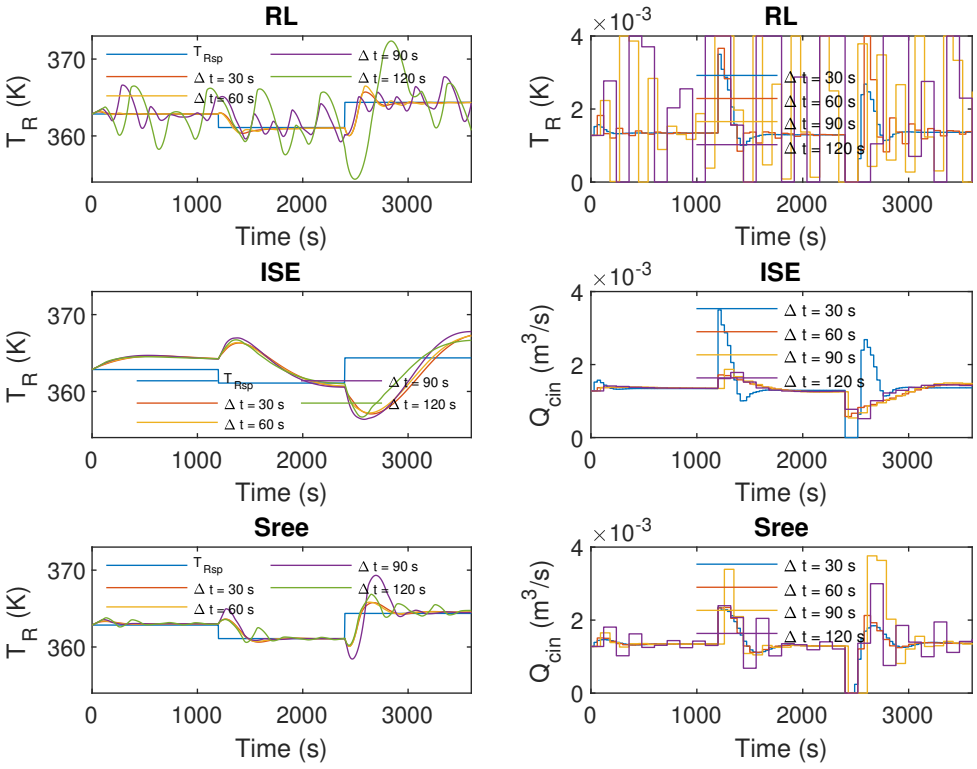


Figure E.16: Temperature closed-loop response for different sampling times