Aix*Marseille université
Socialement engagée

NNT : 0000AIXM0000

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 10 Juillet 2024 par

Anthony KHOURY

Optimization of control algorithms performance in constraint simulation environment: smartphone case

Discipline
Automatique

École doctorale
ED 184 – Mathématiques et Informatique

Laboratoire/Partenaires de recherche
LIS – Laboratoire Informatique et Systèmes

LIS
LABORATOIRE D'INFORMATIQUE & DES SYSTÈMES
UMR 7020

Composition du jury

Pascal LORENZ PR Univ de Haut d'Alsace	Rapporteur
Gregory ZACHAREWITCZ PR IMT Mines Alès	Rapporteur
Thibaut RAHARIJAONA PR Univ de Lorraine	Examinateur
Jaafar GABER MDC-HDR Univ de Technologie de Belfort-Montbéliard	Examinateur
Caroline KULCSAR PR Univ Paris Saclay	Examinatrice
Claudia FRYDMAN PR Aix-Marseille Univ	Présidente du jury
Rachid OUTBIB PR Aix-Marseille Univ	Directeur de thèse
Rafic YOUNES PR Univ Libanaise	Co-directeur de thèse
Invité	
Nazih MOUBAYED PR Univ Libanaise	Membre invité

Affidavit

I, undersigned, Anthony Khoury, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific direction of Rachid Outbib and Rafic Younes, in accordance with the principles of honesty, integrity and responsibility inherent to the research mission. The research work and the writing of this manuscript have been carried out in compliance with both the French national charter for Research Integrity and the Aix-Marseille University charter on the fight against plagiarism.

This work has not been submitted previously either in this country or in another country in the same or in a similar version to any other examination body.

Marseille, 26 avril 2024



Cette œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 4.0 International.

Résumé

Ce travail est dédié à l'optimisation des algorithmes de contrôle sur smartphone. Les défis liés au développement d'applications scientifiques pour smartphones permettant la simulation de contrôle sont d'abord définis en deux catégories principales; utilisation de la mémoire spatiale et calculs scientifiques. Un ensemble de solutions logicielles est proposé et constitue un choix de conception général pour toute application d'ingénierie pour smartphone. Ces solutions sont implémentées dans un prototype d'application de simulation de contrôle conçu en parallèle de ces travaux. Ils sont testés et validés par comparaison avec d'autres applications de simulation standards. Ensuite, l'accent est mis sur des algorithmes spécifiques dans la théorie du contrôle. L'approche est une optimisation mathématique qui cible ces algorithmes. La première cible est la modélisation des systèmes et les algorithmes d'identification en particulier. Ses défis sont d'abord définis sur la base des obstacles explorés dans des travaux antérieurs liés à l'identification et un algorithme optimisé est proposé pour contrer ces défis. Les tests montrent une diminution des coûts de calcul tout en sauvegardant ou réduisant les erreurs d'identification. La prochaine cible d'optimisation est le Processus de Contrôle et particulièrement le Régulateur Quadratique Linéaire (LQR) dont le but est d'évaluer l'impact des techniques de réduction des ressources sur le temps et la précision des calculs. Les méthodes étudiées sont des méthodes de réduction d'ordre, principalement la réduction équilibrée et l'approximation de perturbations singulières. Plusieurs ensembles de simulations sont effectués afin d'évaluer la prévisibilité des performances et des modèles pour le temps de calcul et l'erreur de réduction qui en résulte sont proposés. La réduction équilibrée s'avère être la méthode de réduction optimale dans le cas du contrôle LQR. Ensuite, le même processus de simulation et d'identification est effectué pour analyser l'erreur de réduction. Le résultat est un ensemble de modèles qui représentent la relation entre le gain de temps de calcul et la perte de précision pour LQR. Enfin, le prototype d'application de simulation de contrôle qui a été développé pour tester les solutions proposées est présenté où toutes les sections sont mises en évidence.

Mots clés: Automatique, Simulation, Smartphone, Complexité de Computation, Algorithmes d'Identification, Régulation de contrôle

Abstract

This work is dedicated for System Control algorithm optimization on smartphone. The challenges for the development of scientific smartphone applications that enable Control simulation are first defined as two main categories; Space memory usage and Scientific Calculations. A set of software solutions is proposed and act as a general design choice for any engineering smartphone application. These solutions are implemented in a prototype application for control simulation that was designed in parallel with this work. They are tested and validated through the comparison with other standard simulation applications. Afterwards, the focus shifts towards specific algorithms in Control Theory. The approach is a mathematical optimization that targets these algorithms. The first is System Modelling and identification in particular. Its challenges are first defined based on the explored obstacles in previous work related to identification and an optimized algorithm is proposed that aims to counter these challenges. Tests show a decrease in computational costs while maintaining or reducing identification error. The next target of optimization is the Control Process and particularly the Linear Quadratic Regulator (LQR) where the aim is to assess the impact of resource reduction techniques on calculation time and precision. The methods in study are order reduction methods, mainly Balanced Model Reduction and Singular Perturbation Approximation. Several sets of simulations are conducted in order to assess the predictability of the performance and models for calculation time and resulting reduction error are proposed. Balanced Model Reduction is found as the optimal method in the case of LQR Control. Then the same simulation process and identification is conducted for analyzing the reduction error. The result is a set of models that represent the relationship between calculation time gain and precision loss for LQR. Finally, the prototype application for control simulation that was developed for testing the proposed solutions is presented where all of the sections are highlighted.

Keywords: Automatic Control, Simulation, Smartphone, Computational Complexity, Identification Algorithm, Control Regulation

Nomenclature

Variables

d	Deviation
$\mathfrak{e},\mathfrak{6},e$	Errors
T	Execution Time
m	Input-space size
Ω	Least-Squares Precision Matrix
w	Noise
O	Number of Operations
Q,R	Optimal Control Weight Matrices
K	Optimal State Feedback Gain
Y	Output Matrix
p	Output-space size
φ	Phase at a Given Layer
$\boldsymbol{r}$	Polar Coordinate
r	Radial Order
k	Reduced Matrix Size
δ	Relative Deviation
P	Riccati Equation Solution
N	Sample Size
$\mathcal{T}$	Sampling Period
$\boldsymbol{\sigma}$	Standard Deviation
n	State-space size
x	State-space Variable
X	State-space Variable Matrix
u	System Input
$\mathfrak{Z}$	Total Number of Units
$\emptyset$	Wavefront Phase

Exponent

cor	corrected
S	Data Corresponding to Filtering Iteration s
DM	Deformable Mirror
L	Layer
max	Maximum
min	Minimum
w_{MA}	Moving Average Window
p	Predicted
res	Residual
T	Total
tur	Turbulent

Subscript

BMR	BMR variable
β_1, β_2	Candidate Model Orders
GS	Guiding Star
i	Identification
0	Initial Sample Breakpoint
l	Layer
d	Optical Direction
k	Reduced System size
CPU	Central Processing Unit
s	Sample
SGF	Savitzky-Golay Filter
SPA	SPA variable
L	Total Number of Layers
v	Validation
v	Voltage
$modes$	Zernike Mode

Accent Marks

ΔX	Discrete Derivative
$\widetilde{X}$	Filtered Signal
$\overline{X}$	Mean Value
$\widehat{X}$	Predicted Data
$\overline{\overline{X}}$	Set Median

Abbreviations

AO	Adaptive Optics
AR	Autoregressive
AR	Augmented Reality
ARMA	Autoregressive Moving Average
ARMAX	Autoregressive Moving Average with Exogenous Input Model
ARX	Autoregressive with Exogenous Input Model
BIM	Building Information Modeling
BMR	Balanced Model Reduction
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CLM	Colin-Lindsey-Mallows
CPU	Central Processing Unit
FGLS	Feasible Generalized Least-Squares
FLOPS	Floating Point Operation Per Second
GUI	Graphical User Interface
HAC	Heteroscedasticity and Autocorrelation Consistent
HC	Heteroskedasticity-Consistent
I/O	Input/Output
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
LQR	Linear Quadratic Regulator
LSM	Least-Squares Method
MAF	Moving Average Filter
MCAO	Multi-Conjugate Adaptive Optics
MDE	Model Driven Engineering

MTL	Model Transformation Language
OLS	Ordinary Least-Squares
OS	Operating System
PID	Proportional-Integral-Derivative
PCB	Printed Circuit Board
SA	Suggested Algorithm
SCF	Schur Canonical Form
SGF	Savitzky-Golay Filter
SPA	Singular Perturbation Approximation
SSD	Solid-State Drive
UI	User Interface
WF	Wavefront
WLS	Weighted Least-Squares

List of Tables

Table 1.1: Theoretical number of operations per process 19
Table 2.1: Minimum Memory Requirement for Standard Engineering Applications 36
Table 4.1: Description of Specifications for machines used 65
Table 4.4: Normalised Mean Relative Deviation of Candidate Models 76
Table 4.5: Normalized Mean Validation Error 77
Table 4.6: Mean Relative error during the testing phase 78
Table 4.7: Recorded Mean Error for All Tests 78
Table 4.8: Identification and Validation Error for *BMR* and *SPA* Process 80
Table 5.1: Elevator Control Simulation Parameters 124

List of Figures

Figure 0.1: Computational Balancing in Control Theory 2
Figure 0.2: Thesis Workflow 3
Figure 1.1: Approach for Constraint Simulation Environment 7
Figure 1.2: History of Cellphone Development 8
Figure 1.3: Mobile Application Architecture [20] 9
Figure 1.4: Evolution of Memory and System Requirements 10
Figure 1.5: Memory gap between Matlab requirements and Iphone capacity 10
Figure 1.6: Standard Closed-loop control scheme 12
Figure 1.7: SISO System 12
Figure 1.8: PID Control Concept 14
Figure 1.9: Steps Conducted per Simulation Iteration 22
Figure 2.1: BIM Concept [67] 24
Figure 2.2: AR Application System Architecture [72] 25
Figure 2.3: Medical CAD Application [75] 27
Figure 2.4: Smartphone Scientific Application Challenges 29
Figure 2.5: Combinatorial Optimization Process 30
Figure 2.6: Cloud Computing Architecture [93] 31
Figure 2.7: Standard Burst-Buffer System Architecture [98] 32
Figure 2.8: Shared Memory Parallelism 33
Figure 2.9: UI Design Comparison 35
Figure 2.10: Task 1, 2 Memory Cost Pre-Execution and After-Execution 36
Figure 2.11: Identified Model Save Template 37
Figure 2.12: Conventional Save and Proposed Save Template 37
Figure 2.13: Control Scheme Choice 38
Figure 2.14: Matlab Built-in Examples Memory storage 38
Figure 2.15: Proposed Application Built-in Examples Memory storage 39
Figure 2.16: Control Process Toolbox 39
Figure 2.17: Control Process Task Options 41
Figure 3.1: Challenges for Identification Process 44
Figure 3.2: Centrifugal Pump Concept 45
Figure 3.3: Centrifugal Pump I/O Data 45
Figure 3.4: Variation of FGLS number of operations 46
Figure 3.5: Residual Error of Centrifugal Pump OLS Test 47
Figure 3.6: Relative Error of Centrifugal Pump OLS Test 47
Figure 3.7: Proposed Algorithm Principle 48
Figure 3.8: Observed Variables for Identification Process 51
Figure 3.9: Comparison Process 52
Figure 3.10: RLC Circuit Scheme 53
Figure 3.11: DC Servo Motor Design 53
Figure 3.12: Low-pass filter Scheme 53
Figure 3.13: I/O Data of RLC Circuit 54

Figure 3.14: I/O Data of DC Servo Motor 54
Figure 3.15: I/O Data of Lowpass Filter 55
Figure 3.16: Mean Error Comparison between C1,C2,C3 56
Figure 3.17: Max Error Comparison between C1, C2, C3 56
Figure 3.18: CPU Usage of Literature Methods 58
Figure 3.19: First Comparison with Literature Methods 58
Figure 3.20: RLC Circuit T.2 Mean Error Results 59
Figure 3.21: Variation after Using Generalized Algorithm results 60
Figure 4.1: Planned workflow for the conducted tasks 61
Figure 4.2: Savitzky-Golay Filter of *Riccati* Calculation Time 67
Figure 4.3: Predicted values and relative error 69
Figure 4.4: Derivative Approximation of Filtered Data 69
Figure 4.5: Derivative Approximation Based on Recursive Filtering Output 70
Figure 4.6: Deviation Based on Recursive Filtering 71
Figure 4.7: Variation of Mean Absolute Deviation 72
Figure 4.8: Mean Errors of Candidate Models 73
Figure 4.9: Sample of Data Sets used for Robustness Testing 75
Figure 4.10: Relative Deviation Variation – First Robustness Testing 76
Figure 4.11: Relative Deviation Variation - Second Robustness Testing 76
Figure 4.12: Testing Phase of All Identified Models 77
Figure 4.13: Identification for *BMR* and *SPA* Process 80
Figure 4.14: Simulation Process with Reduction Methods for Fixed Matrix Size 81
Figure 4.15: Filtered Data of Calculation Time with trend lines 81
Figure 4.16: Comparison with Observed Data for Reduction Methods 82
Figure 4.17: FLOPS calculation time sample for Machine A 84
Figure 4.18: FLOPS calculation time sample for Machine B 85
Figure 4.19: Identification of *Riccati* Process on second machine 86
Figure 4.20: Cost Function Approximation Relative Error 89
Figure 4.21: Normalized Logarithmic Sums 89
Figure 4.22: Logarithmic sum and approximations of simulations 90
Figure 4.23: Adaptive Optics Process 91
Figure 4.24: Adaptive Optics Turbulent Phase Closed-Loop 92
Figure 4.25: Testing Phase in Adaptive Optics 96
Figure 4.26: Required Reduced Matrix Size 97
Figure 4.27: Variation of the proposed Cost Function 97
Figure 4.28: Comparison between the different Cost Functions 98
Figure 5.1: Application Software Map 101
Figure 5.2: Application subsection task division 102
Figure 5.3: User Interface Criteria 103
Figure 5.4: Application sub-interface blueprint 104
Figure 5.5: Application Main Menu Divisions 105
Figure 5.6: Application Progress Save Section 105
Figure 5.7: Output Display Example 106
Figure 5.8: Additional Data Display Example 106
Figure 5.9: Model Identification Toolbox Sequencing Example 107
Figure 5.10: Model Identification Toolbox Example 107

Figure 5.11: Control Process Toolbox Map Example 108
Figure 5.12: System Modelling Map 108
Figure 5.13: Parameter Identification Map 109
Figure 5.14: Parameter Identification Application Example 110
Figure 5.15: Parameter Setup Map 111
Figure 5.16: Parameter Setup Example 111
Figure 5.17: System Simulation Map 112
Figure 5.18: System Simulation Example 112
Figure 5.19: Closed Loop Control Map 113
Figure 5.20: Closed Loop Control Example 114
Figure 5.21: Open Loop Control Scheme 114
Figure 5.22: Arduino Lab Toolkit 116
Figure 5.23: Remotely controlled lab kit 117
Figure 5.24: Vehicle transmission built-in example in Matlab Simulink [164] 118
Figure 5.25: Flight control graphical presentation Simulink [179] 120
Figure 5.26: Elevator Reference Control Flow Chart 122
Figure 5.27: Elevator Control Model 123
Figure 5.28: Lab Simulation Design Map 123
Figure 5.29: Elevator Mechanical Design 124
Figure 5.30: Elevator Control Example 125

Table of Contents

Affidavit I
Résumé II
Abstract III
Nomenclature IV
List of Tables VIII
List of Figures IX
Table of Contents XII
General Introduction 1
1 Background 5
1.1 Introduction 5
1.2 Approach for Constraint Environment 6
1.3 Mobile Application Concept 7
1.3.1 Smartphone Dependency Development 7
1.3.2 Mobile Application Architecture 9
1.3.3 Requirements for Simulation Execution 10
1.4 Control Process 11
1.4.1 System/Sensor Modelling 12
1.4.2 Control Design 14
1.5 Computational Complexity 17
1.5.1 Cholesky Decomposition 18
1.5.2 Strassen Algorithm 18
1.5.3 Optimizing Operations for Least-Squares 19
1.6 Simulation Setup 19
1.6.1 Principles Adopted For Simulation 20
1.6.2 Computational Cost Parameters 21
1.6.3 Simulation Environment Setup 21
2 Engineering Smartphone Application Development 23
2.1 Introduction 23
2.2 Overview of Engineering Applications 24
2.2.1 Building Information Modeling 24
2.2.2 Augmented Reality for Visual Enhancement 25

2.2.3 Data Acquisition of Construction Projects 26
2.2.4 Computer Aided Design and Manufacturing 26
2.2.5 Data Collection of Physical Phenomena 27
2.2.6 Discussion 28
2.3 Applications Development Challenges 29
2.3.1 Human / Application Interface 29
2.3.2 Algorithm Execution 32
2.4 Proposed Software Solutions 34
2.4.1 User Graphical Interface 34
2.4.2 Data Management of Input/Output 36
2.4.3 Scientific Calculations 39
2.4.4 Dependency on Programming Languages 40
2.5 Conclusion 41
3 System Modelling: Identification Algorithm 43
3.1 Introduction 43
3.2 Proposed Algorithm Design 44
3.2.1 Challenges for Identification 44
3.2.2 Principle of Identification Process 48
3.2.3 Statement of Algorithm 49
3.3 Comparison of Proposed Methods 51
3.3.1 Considered Process and Used Data 52
3.3.2 Suitable Combination determination 56
3.4 Comparison with Literature Methods 57
3.4.1 Performance of Chosen Methods 57
3.4.2 Comparison with Proposed Method 58
3.5 Generalized Form of Proposed Algorithm 59
3.6 Conclusion 60
4 System Control Simulation Adapted for Constrained Environment 61
4.1 Introduction 61
4.2 Linear Quadratic Regulator Concept 62
4.3 Matrix Order Reduction Methods 63
4.4 Computation Time Modelling 65
4.4.1 Modelling for Riccati's equation solving 66
4.4.2 Recursive Filtering 69
4.4.3 Robustness study 73

4.4.4 Decision of Optimal Model 76
4.4.5 Test Phase 77
4.4.6 Modelling of solving with reduction methods 78
4.5 Model Extrapolation Procedure 83
4.5.1 Procedure 83
4.5.2 Validation Test 85
4.6 Analysis of Precision Loss Vs CPU Time Gain 86
4.6.1 Setup of the relationship cost function 86
4.6.2 Decay Rate Approximation 87
4.7 Case Study 91
4.7.1 Concept of Adaptive Optics Control 91
4.7.2 Closed-Loop Control Process 92
4.7.3 Matrix Dimension Specifications 94
4.7.4 State-Space Control Model 95
4.8 Conclusion 99
5 Prototype Application 101
5.1 Introduction 101
5.1.1 Purpose of Control Application 102
5.2 User Interface Design 103
5.2.1 Main Menu 104
5.2.2 Progress Save 105
5.2.3 Output Display 105
5.2.4 Additional Data Display 106
5.2.5 Dedicated Toolbox 106
5.3 System Modelling 108
5.3.1 Concept of Process 108
5.3.2 Identification of System Characteristics 109
5.3.3 Parameter Setup of Model 110
5.3.4 System Simulation 112
5.4 Control Process 113
5.5 Virtual Lab for Smartphone Simulation 115
5.5.1 Overview on Existing Lab 115
5.5.2 Advantages and Disadvantages 118
5.5.3 Challenges for Simulation Design 119
5.5.4 Elevator Control Simulation 121

5.6 Conclusion.....125
Conclusion and Perspectives.....127
Liste de publications et participation aux conférences 129
References.....130

General Introduction

Smartphones are seeing a rapid increase in spread in comparison to desktops. This increase in reach also shows an increase in dependency on smartphones for regular and daily activities other than just mobile services. The accessibility[1] of smartphones and their future potential have lead researchers to study the possibilities behind mobile technology in several domains such as automating construction site monitoring [1]. However, of the younger and less-explored domains of smartphone applications is in the fields of engineering. Current developments are pushing towards focusing on the smartphone's hardware components or on the smartphone as an outlet of connection such as in cloud-based simulation instead of smartphone-based simulation. One example is Matlab Mobile which is Matlab's mobile application that connects the user to the Mathworks cloud where all operations and data are sent from the mobile to the server to be processed [2]. This in turn leads to the innovation in such applications being limited by the development in hardware/software of smartphones and cloud computing. With the increase in system demands for simulation and modeling, engineering smartphone applications stand currently behind in advancement and innovation in comparison to their PC counterpart.

On the other side of the spectrum, computational complexity[2] in Systems and Control has always been a topic of interest. A problem in this domain can be considered to be satisfactorily solved only if an algorithm is available with moderate computational requirements. The subject is multifaceted and interesting from several perspectives. It can help the practitioner in choosing problem formulations, and in calibrating expectations of what can be algorithmically accomplished. For the systems theorist or the applied mathematician (see [3]), it raises a variety of challenging open problems that require a diverse set of tools from both discrete and continuous mathematics. In turn, for the theoretical computer scientist (see [4]), the problems in systems and control theory provide the opportunity to relate abstractly defined complexity classes and specific problems of practical interest.

As explained in [5], one of the main issues of complexity in Control Theory begins with decision problems, as seen in Figure 0.1, which are considered decidable if there is an algorithm that can determine the correct solution for all instances of the problem in a finite amount of time. In the case of undecidability, several researchers aim to develop the algorithm that offers the closest option to an optimal solution within a feasible time in turn balancing between algorithm precision and execution time. The decidability of several processes in Control Theory has been the topic of several work such as the stability of Saturated Linear Dynamical Systems [6]. However, even in the case of decidable problems, the execution of the correct

[1] Accessibility refers to the ease with which users or systems can interact with and obtain information from a particular simulation or computational model. It also encompasses considerations related to the required resources

[2] Computational complexity refers to the study of the amount of computational resources required to solve a specific computational problem as a function of the input size.

algorithm would result in large resource costs. This has led to the interest in developing approximation algorithms for computation. These are commonly used in optimization where problems cannot be solved exactly in polynomial time. While iterative algorithms optimize memory costs, their implementation can result in increased runtime. This increase is due to the linear execution of operations [7].

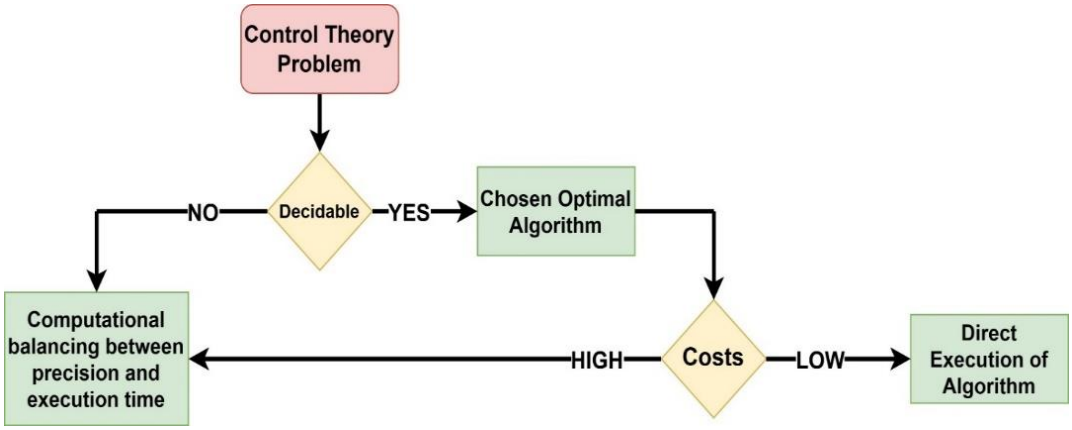


Figure 0.1: Computational Balancing in Control Theory

This work concerns the optimization of control algorithms for the smartphone platform. The primary challenge is balancing between the smartphone's hardware/software constraints and limited simulation capabilities with the computational complexity of Control Theory processes. A large portion of algorithms in this domain are approximation-based where the output has a provable guarantee on the distance of the returned solution to the optimal one [8]. However, reducing this distance i.e. increasing algorithm precision results in larger allocation of execution resources. Hence reducing these resources without mathematical optimization of the algorithm's performance or concerned data allocation would directly result in decreasing output precision. Because of this, the challenge here is reducing the targeted process' allocated resource while maintaining or improving output precision. This is done on two main levels; the first is on a software level where the focus is data management and data interaction and the second is on a mathematical level where the focus is on the performance of the operations involved in the process. Several works in the literature address the issue of computational complexity whether by the redistribution of the workload to a different environment such as in the case of cloud computing [9] or the optimization of the employed resources such as the burst-buffer system [10]. The primary difference in this work is the direct focus on the reduction of the computational complexity.

The main contribution of this work can be summed by increasing the accessibility of complex control simulation processes. This in turn allows the development of control simulation on constraint environments whether it is for educational purposes through virtual laboratories and improving ubiquitous learning or for reducing resource allocation for simulation in a professional scene, therefore increasing efficiency. For theorists, the solutions offered in this work present the ability to increase algorithm efficiency within the constraints of the initial feasible time.

The work begins by exploring the challenges behind the development of engineering applications and complex simulations on smartphone and proposes a set of software solutions that can be implemented for any engineering application from a general perspective. Afterwards, the work focuses on specific domains in Control Theory; System Modelling and Control Process. This is because of the focus on certain standard algorithms that are known for their computational complexity. The approach follows a mathematical optimization process that is specific for each algorithm in concern. Moreover, in order to validate the proposed solutions, a prototype application for control simulation was developed where the observed output can be compared with other known applications for simulation. Figure 0.2 offers a summary of the work conducted.

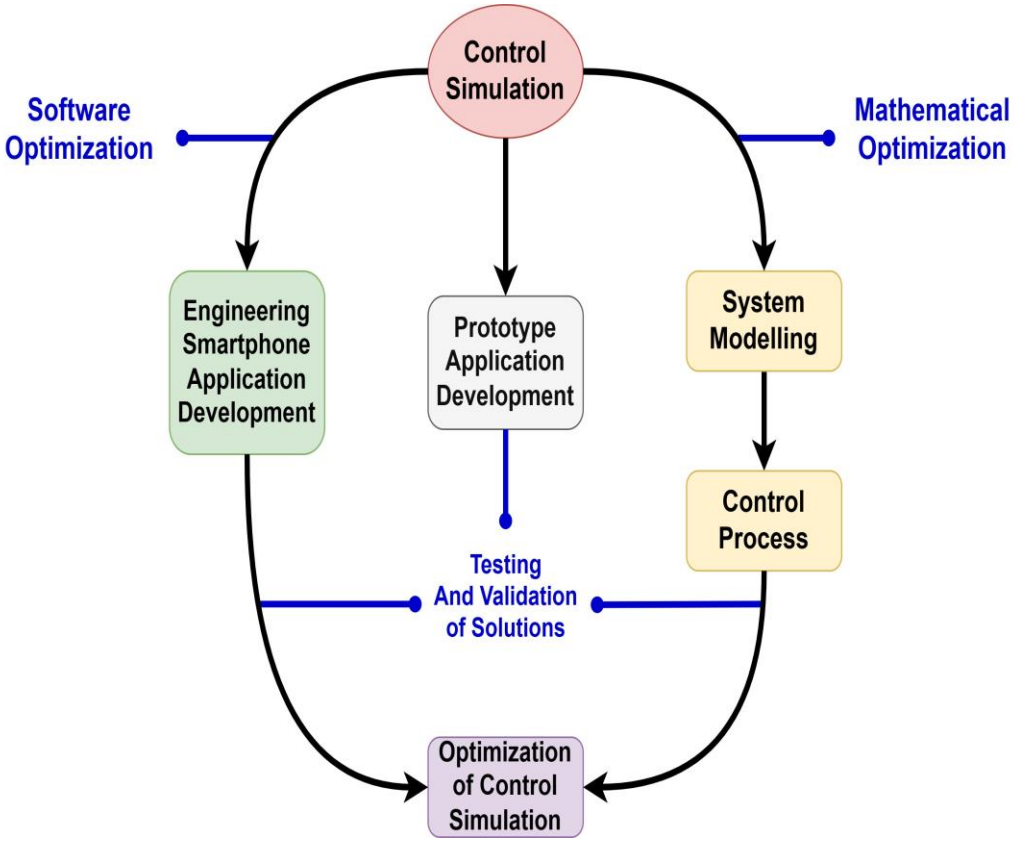


Figure 0.2: Thesis Workflow

The manuscript is organized into five chapters.

Chapter 1 offers a concise overview of the concepts handled in this work, primarily mobile application, control process and computational complexity.

Chapter 2 starts with an overview of standard engineering applications. It then lists the set of challenges that face the development and implementation of engineering applications on smartphones and proceeds to offer an approach to bypassing these challenges through an ensemble of software-based solutions that are presented through the developed control application.

Chapter 3 focuses further on the mathematical optimization of control algorithms. The first target is identification algorithms of linear systems. The work in this chapter focuses on the introduction of an improved identification method to improve the performance of identification algorithms on the smartphone platform.

Chapter 4 also studies the mathematical optimization in the control domain. The work focuses on the LQR control process with model reduction methods. The main objective is to present an efficient model that predicts calculation time and precision loss of LQR process with the implementation of model reduction methods.

Chapter 5 offers a view of the prototype smartphone application for control that was developed for this work. This application showcases the implemented standard methods of control and simulation and also acts as the implementation platform of the developed algorithms.

The final part offers a summary of the results achieved in this work alongside the possible direction of future advancements.

1 Background

1.1 Introduction

The main subject of this work is the target of interest of several domains. In the case of simulation, even the smallest of developed systems require cooperation between engineers of different domains [11]. Concerning smartphones, the design and development of the basis of a mobile application has rapidly increased in complexity because of several reasons including the dependency on several platforms for development in the market [12]. From the perspective of Control theory, there is a constant growth in proposed designs and adaptations in order to increase the efficiency of the control process [13].

This chapter offers an overview of these topics which are related to this work and aims to provide a basis of the operations and concepts of direct interest. The primary objective is defining a constraint simulation environment as a combination of separate domains and the classification of the smartphone platform as such an environment. In turn, the development of smartphones is presented in order to understand their evolution that results in an increase of smartphone dependency in daily activities. Their architecture and design is presented as well.

Computational complexity is also explored to explain the relation between constraint handling and effect on time and space complexity. This approach is inspired by the concept of Constraint Handling Rules[3]. It was developed to address complex constraint problems in a concise and modular way. The main idea behind it is to separate their specification from the control flow of the program [14]. In a similar manner, the first part of this work is defining the constraints related to simulation based on the concepts and philosophies explored in this chapter.

In addition, methods for measuring this complexity and examples of it are also provided with the aim to highlight the difference in each case from a mathematical perspective and the need for targeted optimization when addressing it [15].

This chapter is organized into 3 sections. Section 1.2 defines the concept of Constraint Simulation Environment. The design and development of a mobile application is explored in Section 1.3 alongside the history of smartphone development. Section 1.4 offers further insight into the field of Control.

[3] Constraint Handling Rules which is a declarative programming language designed for writing constraint solvers

1.2 Approach for Constraint Environment

Based on literature, the term “Constraint Simulation Environment” as a general concept is not properly established. However, work targeting the limitations in a given simulation and resulting obstacles from computational complexity is widespread. In the work of Walters et al. [16], the primary challenge for real-time system modelling is the bandwidth of the hosting computer. Another example of this is in [17] where the main motivation for optimization is the limited simulation size on the discussed uniprocessor systems.

As seen in the examples, constraint simulation is targeted in several domains. The main difference is the type of constraint. However, as seen in [18], the main common motivation for working in a constraint environment is the potential for increasing accessibility. This generates from the reduction of required resources which is the primary limitation for operations whether it is because of physical size or cost limit.

In order to analyze further how to approach this type of environment, the Theory of Constraints[4] (TOC) is studied [19]. Despite being a management philosophy, the main objective of TOC is increasing the throughput while respecting constraints. It emphasizes the identification of constraints, which are elements within a system that limit its performance and prevent it from achieving its goals. Once they are identified, TOC suggests exploiting it to its maximum capacity. This involves ensuring that it is always working on the most critical tasks and is not idle due to lack of input or excessive output. Other activities in the organization should be aligned with and subordinated to the constraints. This means that non-constraint activities should support and synchronize with the limitations to optimize overall system performance. If the performance is still suboptimal after exploiting and subordinating, the organization should consider elevating the constraint. This could involve investing in additional resources, improving technology, or making structural changes to remove the bottleneck.

Therefore a constraint simulation environment is characterized by a set of limitations that impact the performance of the targeted operations. These constraints are both the main challenge of operations and the direct result from the primary advantage of the environment which is increased accessibility. In turn, the optimization of the performance of the corresponding simulations is achieved without the increase of allocated resources in order to preserve the required level of accessibility. Figure 1.1 depicts the approach for studying a constraint simulation environment which is dictated by the defined constraints and influences the optimization process.

[4] The Theory of Constraints is a management philosophy that focuses on identifying and alleviating the limiting factors that impede an organization's ability to achieve its goals.

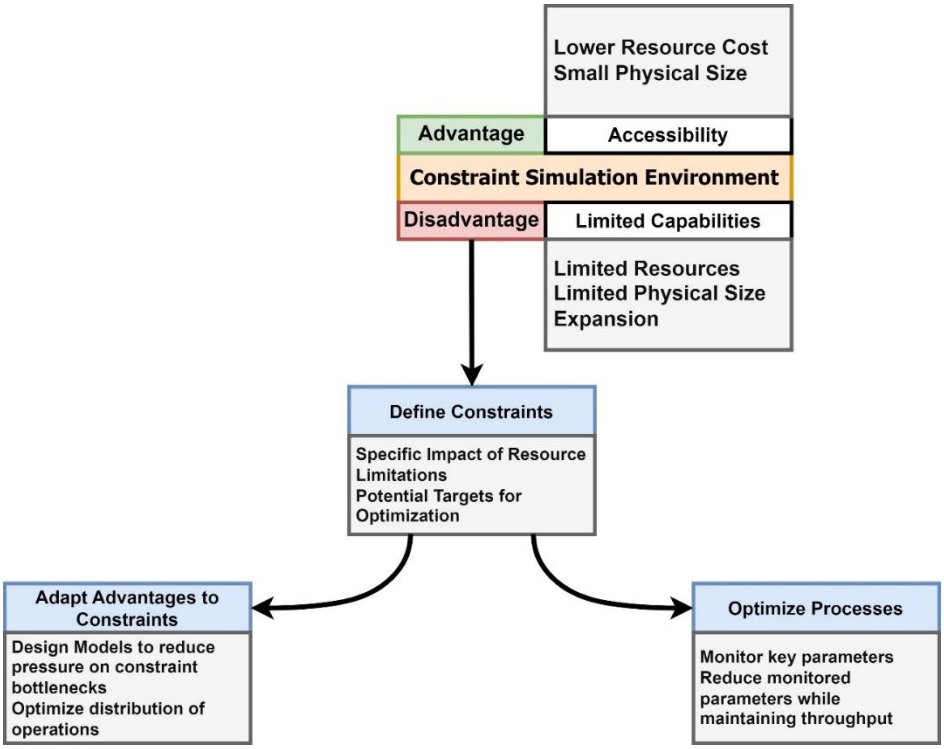


Figure 1.1: Approach for Constraint Simulation Environment

1.3 Mobile Application Concept

In this section, the history of the cellphone is first highlighted to explore the incremental addition of aspects that form the core elements of modern smartphones in addition to defining the concept of mobile application.

1.3.1 Smartphone Dependency Development

Mobile technology has rapidly evolved throughout the years from its initial implementation as a mobile telephony service on the Deutsche Reichsbahn[5] journeying between Berlin and Hamburg to the smartphone we know today.

After several years of mobile telephony service developments, Motorola deployed the first-ever portable mobile device with the DynaTAC 8000X which was first priced at around 4000$ and weighed more than a pound.

In 1992, IBM announced the very first smartphone. It released the Simon Personal Communicator (SPC) for purchase in 1994. The SPC was the first touchscreen phone. Additionally, it had the ability to send and receive both emails and faxes. It also had a calendar, address book, and a native appointment scheduler, and even featured standard and predictive stylus-compatible screen keyboards.

In 1997, the first version of entertainment service appeared on phones with the game Snake that was developed by Finnish engineer Taneli Armanto for Nokia 6110 phones. In 1998, the first downloadable content to mobile phones was released which was the ringtone, launched by

[5] The Deutsche Reichsbahn refers to the state-owned railway company in Germany from 1920 to 1945

Finland's Radiolinja. In 2001, a mobile communications standard was built to allow portable electronic devices wireless access to the Internet [20].

The year 2007 was one of the most influential years in smartphone development and it was when the first Iphone was released. This was primarily because the phone offered users full internet access similar to desktop experience. In turn, 2008 saw the release of the first Android phone T-Mobile G1 alongside the deployment of both Apple's App Store and Android Market, later renamed Google Play Store which became a core part of the modern culture of smartphone dependency [21].

The added features to cellphone throughout the years contributed to the definition of the smartphone that we know today, reshaping mobile technology from an expensive luxury commodity in its beginning to a core part of our daily activities. Figure 1.2 highlights these major advancement in cellphone development.

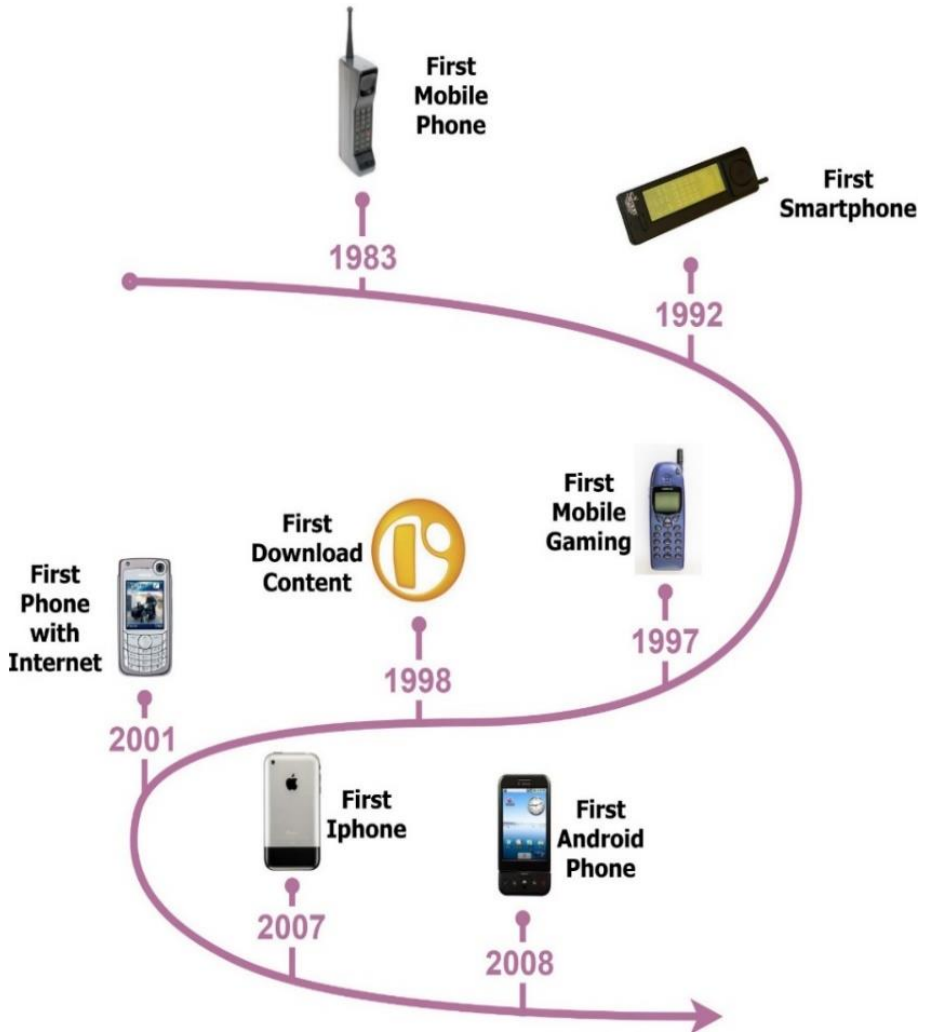


Figure 1.2: History of Cellphone Development

1.3.2 Mobile Application Architecture

The term “application” is used for all electronic platforms and includes both smartphone and desktop. In simple terms, an application is a program that is intended to serve a specific purpose. An application is developed using another that has been designed to fulfill the operation of app designing. The choice of such application to be used is mainly based on the targeted operating system alongside individual preference. Figure 1.3 [22] shows the components of a standard mobile application architecture.

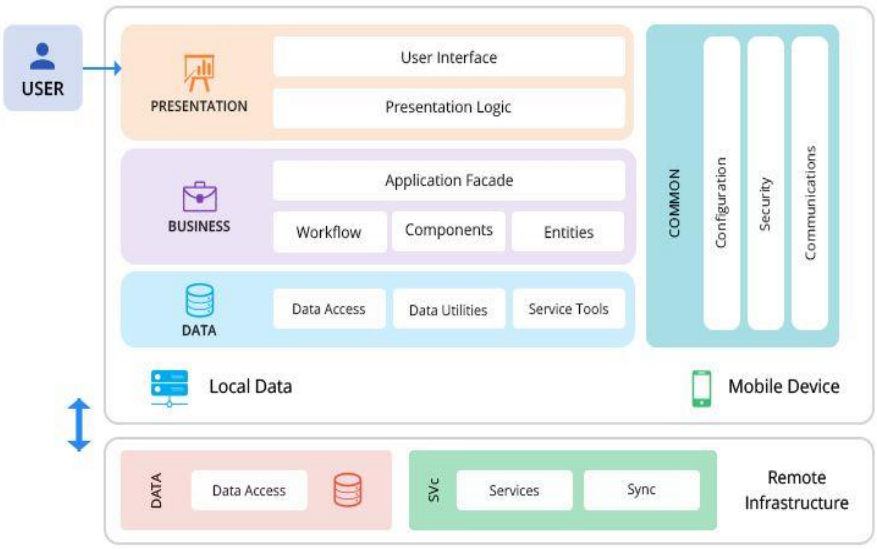


Figure 1.3: Mobile Application Architecture [22]

The Presentation Layer is the front end of the application and represents the main connection between the user and the application. It is composed of UI elements and the related parts [23]. The Business Layer contains the defined business entities alongside the workflows and holds the main connection between the front and back end of the application [24]. The Service Layer is composed of the interfaces where data is translated between the business layer and external data contacts [25]. The Data Layer is formed of the data utilities and the accessed components [26].

Smartphone application tools can be classified in two categories; native mobile application tools (XCode, Android Studio, App Code, etc.) and cross-platform development tools (Xamarin, Ionic, React Native, etc.). The latter category allows the development of an application to be deployed on several platforms.

The two most known operating systems for smartphone are Android and Mac OS, but several other Operating systems have been developed [27]. Complex applications can tend to not only rely on standard design applications for development but on engines which represent a set of designated libraries and assets. Large scale development companies tend to design their own engine to be used for their applications. Based on the previously stated and in broader terms, mobile and desktop applications are similar. However, it is through the analysis of this definition that a smartphone application would differ from its standard desktop counterpart on several aspects other than technical.

1.3.3 Requirements for Simulation Execution

The term "Minimum System Requirements" refers to the set of specifications and hardware capabilities that a certain device must have to run the software or perform a particular function adequately. Meeting or exceeding them ensures that the software can run reliably and efficiently. The minimum system requirements typically include specifications such as processor, memory, storage space, Graphics Card and OS.

In order to visualize the disparity between smartphone hardware and minimum requirements for standard control simulation applications, the Iphone smartphone and Matlab are taken into study as one of the most common in their type. Figure 1.4 shows the evolution of Iphone hardware capacity in comparison to Matlab's minimum required memory throughout the years. The presented data is based on the concerned company's product coverage.

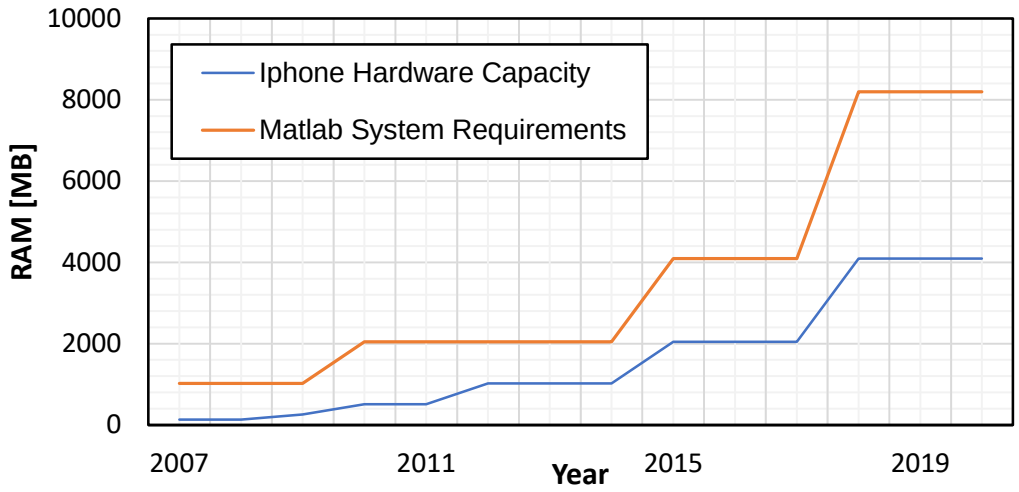


Figure 1.4: Evolution of Memory and System Requirements

Figure 1.5 pinpoints the gap between Matlab's system requirements and Iphone's hardware capabilities.

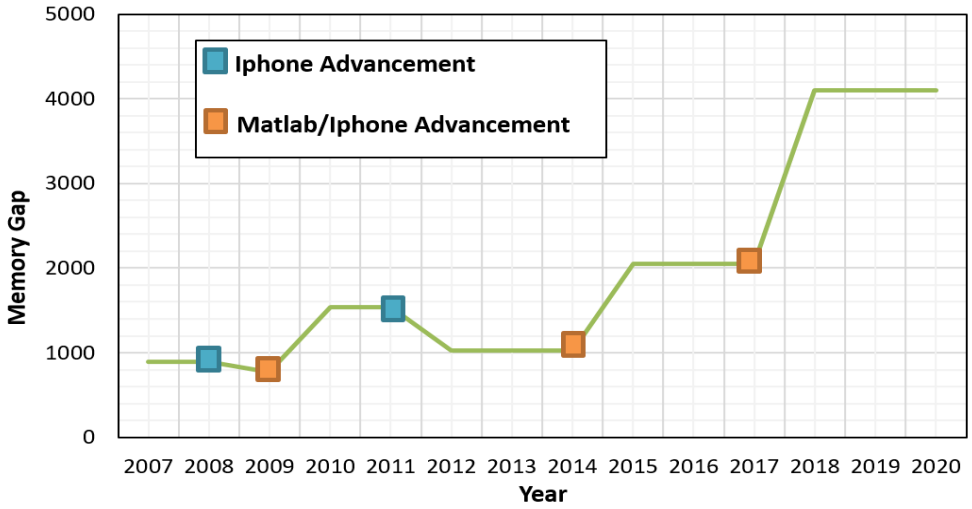


Figure 1.5: Memory gap between Matlab requirements and Iphone capacity

Despite the earlier rapid development of Iphone's hardware capabilities, the difference in memory reached a stagnation that has continued for eight years. The evolution in system requirements and smartphone specification shows an increase in difference throughout the years where an advancement in one is met by an advancement in the other. However, even if minimum requirements are met, the design and integration of smartphone and desktop software differ on several other aspects.

To start on the technical side, desktop applications are developed on the same targeted platform i.e. via other desktop applications, whereas smartphone applications are not developed on smartphone but on desktop alongside the use of emulators[6] and simulators in order to design and test the application [28]. These do not offer fully accurate results because of the high variety between smartphones where small changes in screen size, sensor placements and other can heavily alter the results. Therefore the difference does not only lie between Android and Mac OS.

In addition, the nature of the usage of smartphone dictates that applications are highly preferred to be fast, low downtime and quick results due to the fact that the user might tend to switch around between several applications that are constantly running in the background unlike in the case of desktop where such are usually intended to be solely used for a longer period of time.

This in turn is shown in the difference in the range of operations between desktop and smartphone applications where the latter are usually designed to fulfill a relatively limited purpose in comparison to desktop applications that tend to offer a larger set of operations and options. This is also due to other constraints such as storage size, storage devices which are still grounded by their physical size, thus smartphone storage space remains significantly smaller than that of the standard desktop.

Moreover, while desktop applications must account to certain size restriction as well, desktops offer more chance of customization and enhancement such as purchasing hard disks or SSD, but such enhancements would only hinder the main two features that characterize smartphones which are portability and ease of access.

1.4 Control Process

In this section, the core aspects of the control process are explored which are primarily the design of the system and sensor involved in addition to that of the controller used. The first part of the operation concerns modelling and the second part relates to the operation and conduct of the controller being used and the behavior of the corresponding scheme. While control schemes can vary based on a given or desired scenario, the standard and general design is equivalent to the scheme displayed in Figure 1.6.

[6] An emulator is a software or hardware tool that mimics the behavior of a different computer system, enabling it to run programs or applications designed for the emulated system.

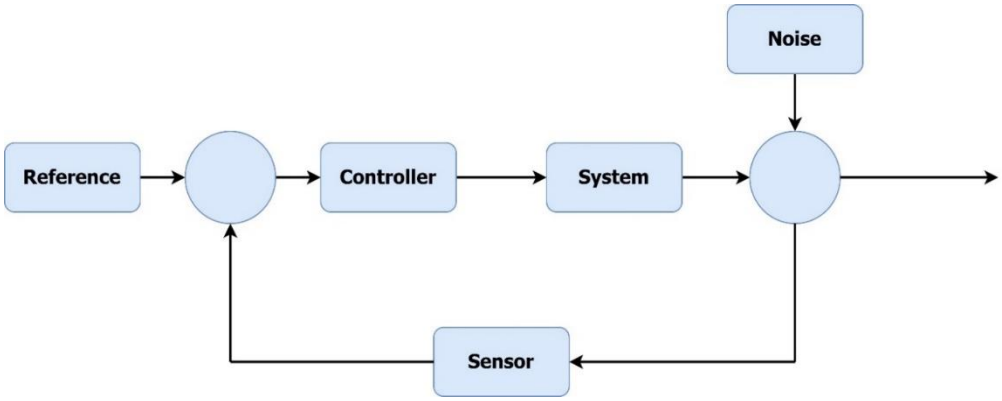


Figure 1.6: Standard Closed-loop control scheme

1.4.1 System/Sensor Modelling

System modeling is the process of representing a real-world system as a mathematical model which is a set of equations that describes the behavior of the system under study. Its purpose is to analyze and design models that can manipulate the system's behavior according to desired specifications.

There are several methods for defining this model, a prominent one is Identification which offers a feasible alternative to physical modeling. Its strength is especially shown when the given system is considered too complex to be physically modeled whether it is because of the theoretical laws involved or other [29]. The work on Identification optimization is widespread and is observed in several domains ranging from control to finance. For these reasons, several algorithms have been developed and introduced. Some of the most common methods introduced for linear identification are the Ordinary Least-Squares (OLS) and Feasible Generalized Least-Squares (FGLS).

1.4.1.1 Ordinary Least-Squares

Considering the linear system in Figure 1.7 with the following form

$$Y = AX \tag{1.1}$$

where $X \in \mathbb{R}^{n \times m}$ denotes the input, $Y \in \mathbb{R}^{1 \times m}$ represents the output, $A \in \mathbb{R}^{1 \times n}$ is an unknown matrix, *n* and *m* are the system's number of inputs and number of observations respectively.

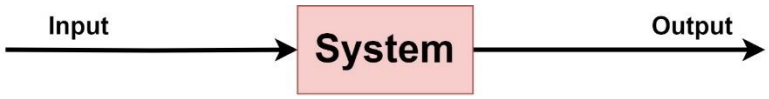


Figure 1.7: SISO System

Based on OLS, the equation for the calculation of A is defined as

$$A = YX^T(XX^T)^{-1} \tag{1.2}$$

The main advantage of OLS is the simplicity of the operations executed. In addition, it preserves data integrity because all data elements are included in the calculation [30]. However, it presents a relative overemphasis on outlier observations or measurements [31].

The OLS is considered the most efficient in certain conditions, primarily if the errors are shown as homoskedastic[7] or normally distributed. In practical cases, the efficiency of this method is measured based on the resulting accuracy. This is because evaluating the homoskedasticity/heteroskedasticity of the errors relies on several observations.

1.4.1.2 Feasible Generalized Least-Squares

In the case of error heteroskedasticity, Weighted Generalized Least-Squares (WLS) is used, an additional matrix is added to (1.2) such as

$$A = Y\,\Omega^{-1}X^{T}(X\,\Omega^{-1}X^{T})^{-1} \tag{1.3}$$

Ω is known as the precision matrix and is the inverse of the weight matrix W. Its mathematical model [32] can be rarely applied because it relies on a large number of replicates at each combination of the predictor variables.

In many practical cases, the weights must be guessed, the analysis is performed then they are re-estimated based on the results and several iterations may be necessary [33]. The use of WLS leads to the need of more stringent assumptions in comparison to OLS. It requires a rarely satisfied assumption that the exact form of heteroskedasticity is known.

As an alternative, the FGLS can be applied [34] which is as follows

1. Identify the system via OLS.
2. Estimate a model explaining the squared residuals from the estimation.
3. Identify the system again, this time minimizing the weighted sum of squared residuals, where the weight is found as the inverse of these residuals.

The approach above shows the needed estimation when conducting WLS even with the feasible route. Most of the standard approaches for W rely on the study of residuals. In standard, the model of the weight tends to be suggested by economic theory, exploratory data analysis or by convenience [35]. Several FGLS methods have been developed and altered throughout the years.

The heteroskedasticity-consistent covariance estimator is one of the techniques used to address the issue of heteroskedasticity introduced by White [36] in addition to several iterations of it [37] alongside further development by Cribari-Neto [38].

Other methods were developed later on such as the Colin-Lindsey-Mallows (CLM) method [39] which adjusts the White estimator for cross-sectional dependence by including a correction term in the variance formula. Another standard method, Heteroscedasticity and Autocorrelation Consistent Method (HAC) [40], was introduced and follows the Newey-West equation for the calculation of the precision matrix.

[7] Homoscedastic refers to a statistical concept related to the uniformity or constancy of variability in a set of data points. In the context of regression analysis, it means that the spread or dispersion of the residuals is consistent across all levels of the independent variable. It is the opposite of Heteroscedastic

1.4.2 Control Design

A controller is a fundamental component in control systems engineering used to regulate and manage the behavior of a dynamic system to achieve desired objectives. It plays a crucial role in maintaining or optimizing performance by adjusting control inputs based on feedback from the system itself or external references. A controller is primarily designed to regulate a system's behavior. It ensures that it operates within desired performance specifications and maintains certain variables or outputs at desired setpoints.

1.4.2.1 Feedback Based-Control

These processes use feedback from the system's output (the measured process variable) to adjust the control input. They aim to maintain the system's output close to a desired set-point or reference signal. A known example of this type of control is the PID controller, as seen in Figure 1.8, which includes the proportional controller, the proportional plus integral controller, the proportional plus derivative controller and the PID controller [41]. They are widely used in the industry because of their simple structure, ease of tuning and implementation [42]. Their simplicity primarily derives from being a reactive controller where in practical implementations, the parameters can be calibrated online based on recorded observations. In addition, the tuning of each parameter follows a clear reactive approach.

The proportional term K_p considers the current error between the desired setpoint and the actual process variable. It applies a correction to the output that is directly proportional to the current error. A higher K_p value leads to a stronger corrective action, but it can result in overshooting or oscillations. The integral term K_i looks at the accumulation of past errors over time. It applies a correction based on the integral of all past errors, aiming to eliminate any residual steady-state error. The integral action helps eliminate long-term steady-state errors. It is effective when there is a constant disturbance that the proportional term alone cannot eliminate. The derivative term K_d considers the rate of change of the error. It applies a correction based on the prediction of future error by looking at how fast it is changing. This helps dampen the response and reduce overshooting. It anticipates the future trend of the error and acts to counteract abrupt changes, providing stability to the system.

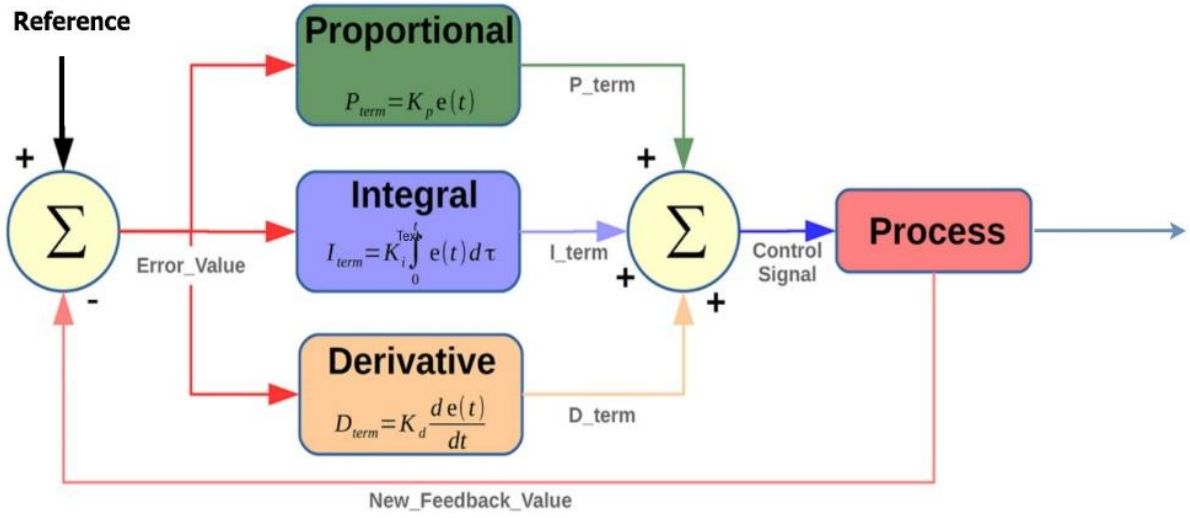


Figure 1.8: PID Control Concept

1.4.2.2 Optimal Control

Optimal control is a category of control methods that aims to optimize a specific performance criterion while considering system dynamics and constraints. This optimization is based on the minimization of a set cost function. Known examples of Optimal Control include the Linear Quadratic Regulator (LQR) [43] and Linear Quadratic Gaussian (LQG) [44].

The process of designing LQR is a well-established problem in control theory, extensively covered in standard textbooks. It involves determining the state feedback gain that optimizes a quadratic cost function related to the plant's state and input variables. This gain calculation necessitates solving a Riccati equation and is expressed as a function of the plant's state-space model. The matrices *Q* and *R* play a major role in shaping the behavior of the controller. *Q* defines the weight of state variables in the function and *R* defines the cost associated with control efforts. By tuning these matrices, the controller's performance and behavior are adjusted to meet specific control objectives. In essence, it relies on a sufficiently accurate explicit model of the plant and employs it in the control design process, adhering to the certainty equivalence principle [45].

Consider the discrete system with input u, output y and state-space variable x such as

$$\begin{cases} x(i+1) = Ax(i) + Bu(i) \\ \quad y(i) = Cx(i) + Du(i) \end{cases} \tag{1.4}$$

where $i \in \mathbb{N}^*$, $A \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{p\times n}$ are the state space parameters and $B \in \mathbb{R}^{n\times m}$, $D \in \mathbb{R}^{p\times m}$ are input matrices, $with\ n, m, p \in \mathbb{N}^*$

The infinite horizon LQR control problem can be summarized as finding the optimal state feedback gain K of the control law

$$u(i) = -Kx(i) \tag{1.5}$$

With the quadratic cost function

$$J = \sum_{i=0}^{\infty} x(i)^T Q x(i) + u(i)^T R u(i) \tag{1.6}$$

The optimal gain is given by

$$K = (R + B^T P B)^{-1}(B^T P A) \tag{1.7}$$

P is the unique positive definite solution to the discrete time algebraic Riccati equation

$$P = A^T P A - (A^T P B)(R + B^T P B)^{-1}(B^T P A) + Q \tag{1.8}$$

1.4.2.3 Adaptive Control

Adaptive control involves the automatic adjustment of control parameters based on real-time changes in the system dynamics. Unlike traditional fixed-parameter controllers, it continuously adapts and optimizes its parameters to account for variations, uncertainties, and changes in the controlled process.The primary goal of adaptive control is to enhance the performance, stability, and robustness of a system in the presence of dynamic and uncertain environments. It achieves this by monitoring the system's behavior, identifying variations or disturbances, and adjusting its control strategy accordingly. This adaptability is particularly beneficial in situations where the system's characteristics may change over time, or when precise mathematical models are challenging to develop due to uncertainties. An example of this is the work of Patino et al. [46] where an approach to model reference adaptive control based on neural networks is proposed and analyzed for a class of first-order continuous-time nonlinear dynamical systems.

1.4.2.4 Robust Control

Robust control is a branch of control theory that focuses on designing systems that can maintain satisfactory performance and stability in the presence of uncertainties, variations, and disturbances in the system being controlled. Its primary goal is to ensure that a control system performs well across a range of operating conditions and in the face of uncertainty. The H-inf controller is considered a robust technique because its primary focus is on designing controllers that provide stability in the presence of uncertainties and disturbances. It is a specific type of robust control method that operates through the minimization of the H-inf norm of the system transfer function, which represents the maximum gain from disturbances to controlled outputs [47].

1.4.2.5 Model Predictive Control

Model Predictive Control (MPC), also known as Receding Horizon Control or Moving Horizon Control, is an advanced control strategy used in a wide range of applications, including process control, robotics, automotive control, and more. MPC is unique because it considers a prediction of future system behavior while optimizing control inputs over a finite time horizon. Improvements in the industry led to the development of a more general model based control methodology in which the dynamic optimization problem is solved online at each control execution. Process inputs are computed so as to optimize future plant behavior over a time interval known as the prediction horizon. In the general case any desired objective function can be used. Plant dynamics are described by an explicit process model which can take, in principle, any required mathematical form [48].

1.5 Computational Complexity

Computational complexity refers to the study of the amount of computational resources, such as time and space, that an algorithm or a problem requires as a function of the size of the input. In simpler terms, it's a way of analyzing and categorizing how the efficiency of an algorithm or the difficulty of a computational problem scales with the size of the input. Space and time complexity are considered two of the main measuring spectrums when assessing memory consumption and computational time.

Giving a theoretical accurate value for memory consumption is not usually sought after due to the intricacies that vary between each device and software capacity that can drastically change between a manufacturer and another, both in hardware and software [49]. For such reasons, methods and studies of approximation or setting upper and lower boundaries of space complexity are adopted for evaluation [50]. Even though each mathematical basic function such as sum are practically instantly executed, its cascade in a significantly large number of loops and the required number of variables means that even the smallest of changes in an algorithm can cause an exponential growth in costs. For such reasons, measuring the complexity of an algorithm does not involve only an approximate of memory cost but the measurement of order-of-growth (Big O) as well which specifies the growth in memory cost in relation to any possible change in variable and parameter size [51]. In addition, while algorithmic complexity directly affects execution time which in turn heavily affects user experience, another possible outcome is full memory usage. This can directly lead to a freeze of operations and software crash and possibly, hardware damage.

The order of growth is an important factor for programmers and mathematicians alike and tends to be employed as a standard order of approximation. It is a mathematical notation that represents the limiting behavior of a certain function when its input tends towards infinity or practically to a particular maximum value [52].

Computational complexity of operations is a topic of interest for the smallest and most basic of operations such as matrix multiplication to advanced and large processes such as identification algorithms and LQR control which are addressed here later on.

An example of this is the previously explained OLS i.e. (1.2), considering the number of operations for computing based on native processes as

$$O(X^T) = n\,m,\ O(XX^T) = m\,n^2,\ O((XX^T)^{-1}) = \frac{5}{6}\,n^3 \text{ and } O(YX^T) = m\,n$$

Then

$$O((1.2)) = m\,n^2 + \frac{5}{6}\,n^3 + m\,n + n^2$$

Several algorithms were proposed to reduce the number of operations used by Matrix Inversion and Multiplication, primarily Inversion through Cholesky Decomposition and Strassen Algorithm respectively.

1.5.1 Cholesky Decomposition

Considering a positive-definite Hermitian matrix[8] $A \in \mathbb{C}^{n\times n}$, the objective of the Cholesky Decomposition [53] is to obtain F, an upper triangular matrix, such that

$$A = F^T F$$

Considering X$\in \mathbb{C}^{n\times n}$ such that

$$AX = I \Leftrightarrow F^T FX = I$$

Set

$$FX = B \Leftrightarrow F^T B = I \Leftrightarrow B = F^{T^{-1}}$$

A matrix S is constructed such that

$$s_{ij} = \begin{cases} \frac{1}{l_{ii}} & if\ i = j \\ 0 & if\ i \neq j \end{cases}$$

Backward substitution is used to solve for x_{ij} using $Fx_i = s_i$

1.5.2 Strassen Algorithm

Strassen algorithm [54] offers an alternative for the classical method of matrix multiplication. Consider the following matrices A, B, C as $n \times n$ matrices where

$$C = A\,B\,, n = 2^h\,, h \in \mathbb{N}^*$$

A, B, C can be split into a set of sub-matrices such that

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \qquad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

The native algorithm defines C as

$$[C_{11}\ C_{12}\ C_{21}\ C_{22}\ ] = [A_{11}B_{11} + A_{12}B_{21}\ \ A_{11}B_{12} + A_{12}B_{22}\ \ A_{21}B_{11} + A_{22}B_{21}\ \ A_{21}B_{12} + A_{22}B_{22}\ ]$$

Strassen's algorithm defines the following elements

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22}) \quad M_2 = (A_{21} + A_{22})B_{11} \quad M_3 = A_{11}(B_{12} - B_{22})$$
$$M_4 = A_{22}(B_{21} - B_{11}) \quad M_4 = A_{22}(B_{21} - B_{11}) \quad M_5 = (A_{11} + A_{12})B_{22}$$
$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12}) \quad M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

Then

$$[C_{11}\ C_{12}\ C_{21}\ C_{22}\ ] = [M_1 + M_4 - M_5 + M_7 \quad M_3 + M_5 \quad M_2 + M_4 \quad M_1 - M_2 + M_3 + M_6\ ]$$

[8] A Hermitian matrix, in linear algebra, is a complex square matrix that is equal to its own conjugate transpose.

These steps are recursively repeated until A_{ij} and B_{ij} are elements and not matrices. Note that this algorithm can be applied to non-square matrices where $n \neq 2^h$ as well where extra-elements are filled with 0 until the criteria are achieved.

1.5.3 Optimizing Operations for Least-Squares

Table 1.1 explains the theoretical number of operations needed for each of the methods in concern for $n \times n$ matrices

Method	Number of operations
Cholesky Decomposition Matrix Inversion	$\frac{1}{2}\, n^3$
Direct Matrix Multiplication	n^3
Strassen Algorithm	$7\, n^{2.81} - 6\, n^2$

Table 1.1: Theoretical number of operations per process

Returning to (1.2), the modified number of operations becomes as follows

$$O((1.2)) = m\, n^2 + \frac{1}{2}\, n^3 + m\, n + n^2$$

Based on the presented processes for each operation optimization, several multiplication and inversion cannot be altered because of matrix dimension conditions. These restrictions forbid the application of these types of general optimization for several processes in Control. This results in the need for a targeted mathematical approach that specifically addresses each specific process individually.

In addition, while (1.2) represents a single process, other algorithms such as in (1.8) rely on several sets of iteration until convergence where the specific number of iterations is unknown therefore we are unable to define the number of operations for the entire process. Because of this, different approaches are taken for measuring costs and achieved degree of optimization such as CPU Calculation Time and CPU usage.

1.6 Simulation Setup

In several areas of work, the majority of published research depends on experiments to support their conclusions. The capacity to draw similar results by replicating an experiment conducted by other researchers is crucial to reach a consensus on an empirical claim. From an engineering standpoint, the assumption that experimental results hold under similar conditions is vital for making informed decisions and predicting outcomes when addressing real-world problems [55].

A large portion of the work conducted here relies on a heavy set of simulations for data extraction and result observation. In this section, the main principles for conducting simulation

are explained alongside the numerical values that are considered as the threshold for a successful simulation.

1.6.1 Principles Adopted For Simulation

In the literature, authenticity and validity of a presented simulation is assessed through the ability to re-conduct these simulations on other independent processors and machines (see for instance [56] [57] [58]). Four core pillars are considered when approaching simulation validity based on the Turing Principle [59]. The two distinguishing factors that differentiate between these pillars are the dataset used and the executed method for analysis and whether or not they are similar or different to that used in the original simulation, resulting in these four different approaches shown in Table 1.2.

	Dataset	**Analysis Method**
Reproducibility	Similar	Similar
Replicability	Different	Similar
Robustness	Similar	Different
Generalisability	Different	Different

Table 1.2: Turing Principles of simulation recreation

The difference between the combination of the dataset and analysis method allows the testing of the conducted simulation from varied perspectives and in the case of failure in validation, this allows an educated guess of the possible factors contributing to this deterioration in performance. Apart from the setup of the simulation environment, a large contributing factor to the efficiency of the adaptation of the models to different processors is the software implementation. While the domain of replication and reproducibility is vast and varying between different fields, a set of broad design choices [60] are to be followed.

Hierarchal Design. The simulations in concern must be designed primarily from reusable components i.e. global and local variables as well as defined functions. This minimizes the need for any extra-memory allocation which can alter the performance of the CPU.

Modular Design. The used functions and iterations must follow a modular approach where they can function both separately and collectively. This is to ensure a linear approach to the simulation and in turn a sequential set of separate iterations.

Repeated and Distanced Testing. Simulations are run on the same model with the same parameters where several simulations are needed for every observed value. This is important in the case of CPU Time observation because of the lack of verifiable consistency of the observation from a single simulation. The average of a large sample set per observed instance would aid in minimizing the effect of external factors on the observation. In addition, it is preferred to conduct these samples at different times and not sequentially in order to further minimize the effect of any temporary and not instanced perturbation in the observation.

In the two forthcoming subsections, the framework of the simulation work is defined taking into account the principles described above.

1.6.2 Computational Cost Parameters

In this work, two main parameters are used for measuring computational complexity, namely Processer Usage i.e. $\mathcal{U}_{CPU}$ and Processor Calculation Time i.e. T_{CPU} where they are respectively used in Chapter 3 and Chapter 4. Different parameters are selected for each section of the work primarily because of the processor performance where in the case of the Least-Squares Identification work, the required computations do not saturate the capabilities of the processor in turn the complexity is measured based on the ratio of processor capacity used whereas in the work concerning the Riccati process, processor saturation for a quantifiable time leads to resulting to a different measurement. As explained previously, the measurement of these parameters requires repeated testing where the presented results are the mean of a large set of successful trials. The need for repeated testing is mainly because of the variation in processor performance between trials. Apart from potential deterioration in performance, this variation can be primarily accredited to dynamic hardware adaptation in manufactured processors. These adaptations offer a variable performance in order to maximize efficiency within voltage usage and temperature constraints [61].

In a similar fashion, numerical results of accuracy mainly relative error measurements also represent the mean of several trials when the methods used are iterative and rely on initial estimates before conducting iterations. This is to verify the convergence of the employed methods to a constant value regardless of initial estimates after a sufficient number of iterations is conducted.

1.6.3 Simulation Environment Setup

In order to ensure the success of a simulation test based on the previously stated Turing principles, a set of parameters are both chosen and observed. In the case of processor performance observation, the main contributing factors for simulation efficiency are the ***Confidence Interval*** and the ***Number of Simulation Iterations Conducted***. For the first, a successful observation test for extracting data must show a maximum standard deviation of five per cent in comparison to the set mean data set which is based on several sets of simulation conducted. The numerical choice is based on a conventionally chosen basis in standard statistical tests [62]. Concerning the number of simulations, this varies between each chapter depending on the complexity and size of the conducted simulations as well as the main observed results. In addition, a large enough number of simulations is chosen also in the case of iterative methods in order to ensure proper convergence to a stable output regardless of initial estimation. Table 1.2 shows the number of simulations conducted per section of the work.

	Number of Simulation Iterations
Identification Process	10^3
Riccati Process	$5\ 10^3$
Standard Matrix Operations	$2\ 10^5$

Table 1.3: Chosen Number of Simulations per Process

The primary difference between the three processes is primarily due to the characteristics and objectives of each one. For the ***Identification Process,*** the main observed result is the mean relative error alongside the processor usage. Concerning the latter, the observed output is a validation of the design choices in the proposed work that ensure in principle a reduction in computational complexity. Concerning the ***Riccati Process,*** **t**he primary parameter of observation is the Calculation Time where the largest section of the work focuses around it. Therefore a large number of simulations is chosen while balancing between the precision of the observations and the feasibility of obtaining the data needed for the work in a balanced time. As for ***Standard Matrix Operations***. The number of simulations conducted here is the largest simply because of the noticeably smaller complexity of the operations in comparison to the other processes, therefore allowing a much larger number of tests in a smaller time frame. Note that each of these simulations is conducted at similar circumstances to minimize the possibility of any external factor influencing the output which is through the application of the previously set pillars through the clearing of memory cache and other steps alongside hardware monitoring through processor temperature observation or other. Figure 1.9 summarizes the steps for the simulation process.

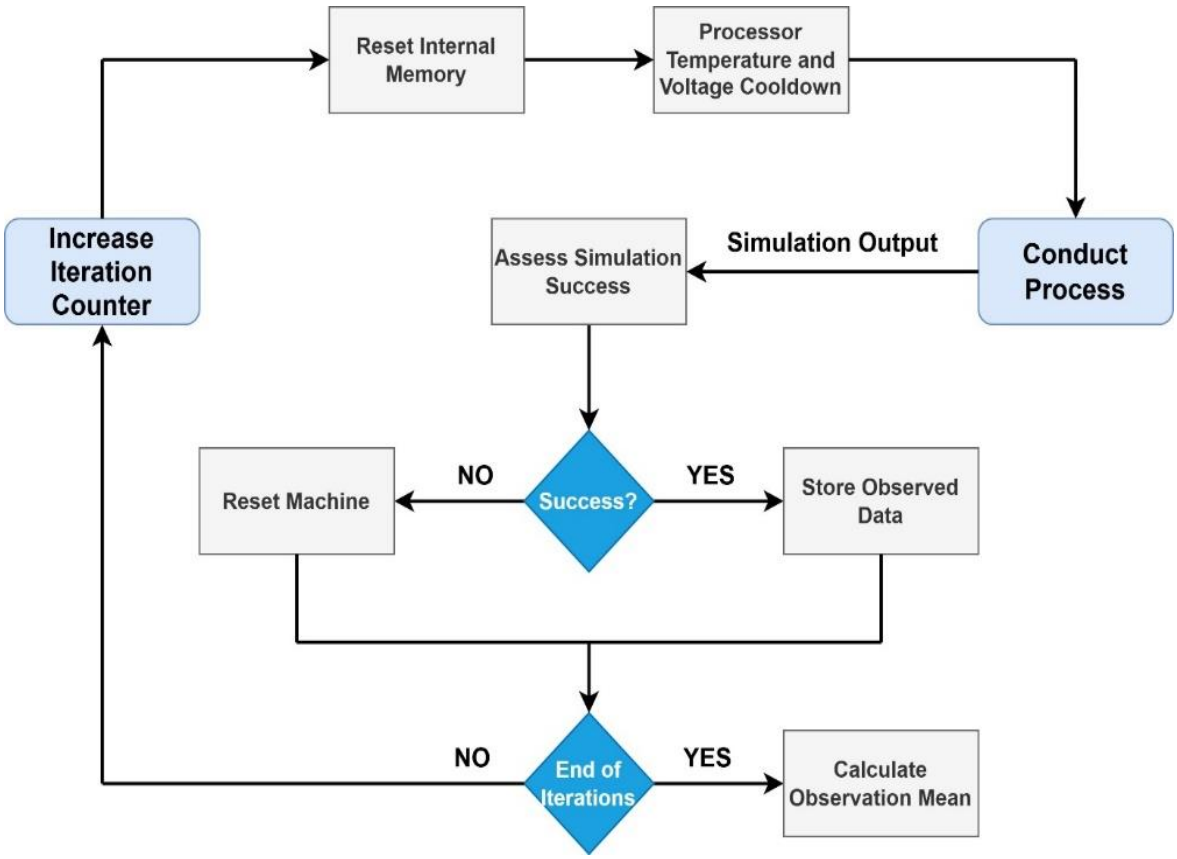


Figure 1.9: Steps Conducted per Simulation Iteration

2 Engineering Smartphone Application Development

2.1 Introduction

The following chapter focuses on defining the challenges of developing and implementing an engineering application on the smartphone platform. The introduction of smartphone usage to a new domain is primarily accompanied by several implications, mainly incorporation of non-traditional interactions, interfaces, and environments [63]. These types of applications have been rapidly increasing in size and scope for the past several years but have yet to reach the level of their desktop counterparts. While there are several variables that contribute to the emergence of the detailed challenges later on, the primary fact is the limitations in the smartphone's hardware/software capacity. In addition, the most known and standard engineering applications operate on engines that are directed to the desktop platform which in turn greatly increases system requirements and limits the possibility of integration on the smartphone platform.

Other than the challenges related to the smartphone, the design of scientific applications, regardless of the platform, tends to require a specific approach that varies between domains. This is apparent in the variation between the published work throughout the varied field such as in spectral analysis [64], radiogenomic studies [65], agriculture [66] and other. This difference is primarily due to the type of data that is to be presented, industry standards and conventions, displayed practiced and other.

The methodology for the work presented in this chapter begins by an overview covering standard smartphone engineering applications. Afterwards, a set of challenges is conceptualized. The developed application is used as a case study to offer a functional presentation of the solution for each of the explained challenges. The main contribution is to provide a set of possible challenges that should be considered from the start of the development of an engineering smartphone application alongside a solution example and in turn allow future work to efficiently bypass these challenges.

The work in this chapter is split between three main sections. Section 2.2 offers an overview of engineering applications. The challenges in the development of engineering applications are defined in Section 2.3 and Section 2.4 shows the corresponding software solutions.

2.2 Overview of Engineering Applications

In this section, the professional use of smartphone applications in the field of engineering is explored. While there are several other applications for specific domains, the focus here is on applications types that have seen several variations and iterations in the industry.

2.2.1 Building Information Modeling

In terms of the context of construction, a prominent feature of mobile applications is augmented reality where the accessibility of smartphones and future potential have lead researchers to study the possibilities behind mobile technology for automating construction site monitoring [67]. On the other side, the advantage of using Building Information Modeling (BIM), shown in Figure 2.1, is its ability to offer consistent and coordinated views alongside a representation of the 3D model that is supported by reliable 4D representation which is time and 5D data which is cost for design, construction and operation of built assets have been widely praised for their advancement and operation [68].

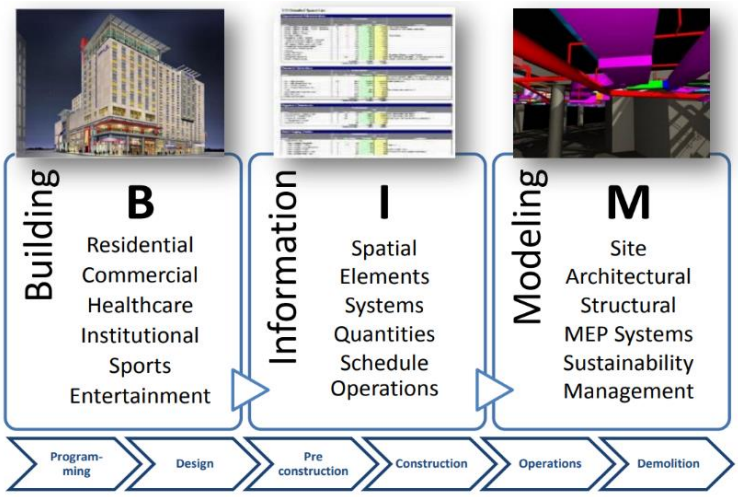


Figure 2.1: BIM Concept [69]

Such benefits would result in the improvement of the overall productivity of onsite operations due to the introduction of new project tracking abilities as well as facilitating interactions and data sharing through the participants in real-time. Efficient project tracking is dependent on periodic reports that are traditionally based on manual information collection with frequent transcription. Previous and current work have focused on exploring the potential behind mixing between real environment and virtual information. One of the important benefits that is sighted is the possibility of reducing the time and cost behind explanation and design description which hinder the efficiency of conducting value-adding tasks where studies show a general low effectiveness rate on explanation and description tasks [70]. This is due to an eminent lack of precision in explaining the drawings and design of an individual to others which can result in design defect and construction rework. BIM is a digital representation of a building and its internal features. It contains intelligent construction components that can include data attributes and parameter rules per item.

This modeling process has attracted worldwide attention in Architecture, Engineering, Construction and Resource Management as there is growing evidence that adopting it increases efficiency and productivity [71]. BIM architecture management and editing tools are widely used in conflict detection, model and location communication and planning. Some of the notable BIM smartphone applications include AR Mapper, Revizto, ARKi and BIMx.

2.2.2 Augmented Reality for Visual Enhancement

Augmented Reality (AR) is a growing technology that is based on the coexistence between live and real images and virtual information through the mobile interface medium. This first appeared in the fields of retail, travel, entertainment, social communication and advertising but its increased affordability and ease of use has made the implementation of the technology in the construction industry more feasible and approachable [72].

Recent AR applications use different tracking configurations that can be categorized into 'mark-based' and 'marker-less' types. Marked-based tracking is used to track a point that mobile cameras can reliably detect. In addition, the design of the marker can often ensure rapid alignment to allow for effective tracking.

On the other side of the spectrum, marker-less tracking settings can set and track different targets without any markings and allow the use of Global Positioning System, shapes, face / image detection, 3D maps, and more [73]. AR implementation is still a relatively new field in construction but some applications have already been present such as GAMMA AR, Morpholio AR and DAQRI Smart Helmet. Figure 2.2 shows the architecture of an AR smartphone application.

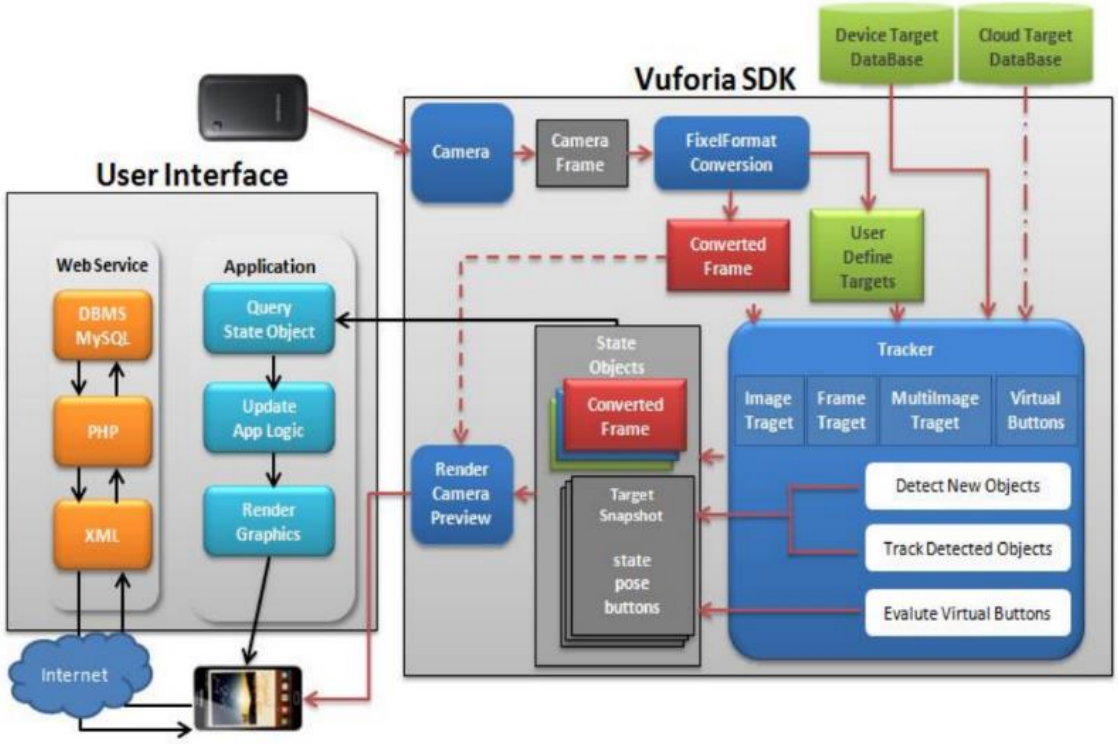


Figure 2.2: AR Application System Architecture [74]

2.2.3 Data Acquisition of Construction Projects

A study from 2012 in the USA showed that 87% of subcontractors and 93% of the general contractors sampled were utilizing mobile devices on job sites in order to boost productivity where such uses typically range from the basics (like cameras) to more technologically advanced features like GPS and sensing devices. The practice of progress tracking heavily relies on supervisors' regular reports that depend on intensive manual collection of data which can lead to data entry errors [75]. Field engineers utilize 2D planned drawings, construction details and project specifications in order to review the current progress thus far. Afterwards, the construction schedule is studied with the aim of identifying the work that is planned to be operated. With such a large amount of manual work, there is always the risk of affecting progress estimations.

With the aim of overcoming these limitations, commercial applications have been developed in order to improve construction monitoring alongside detailing progress on the site. One example of such is 'Site Progress'.

2.2.4 Computer Aided Design and Manufacturing

Computer Aided Design (CAD) represents the technology that is related to the implementation of computer systems to assist in the creation, analysis and modification of a design. The framework of a smart CAD model that consists of the collaborative design protocol which is reinforced by the design history structure has constantly shown development. Apart from the general challenges that face the implementation of standardized computer applications on mobile devices, applications related to CAD have originally struggled in achieving the expected success and approval rate on smartphones due to the restrictions that come with a relatively small screen, touch screen precision deficiency and general lack of modeling power on the smartphone software. However, with the increase in development and complexity of smartphones, CAD mobile applications have seen a rise after a somewhat weak start [76]. This can be considered partly due to the developments of tablets more so than smartphones due to the tablets' advantage in screen size and general sense of bureaucratic work. Therefore the increase in interest in tablet CAD applications gradually shifted to the smartphone as the latter showed further development and innovation.

However it is to be noted that the popular use of smartphone CAD applications is for reviewing and modifying previously made designs that were created on computers and desktops, therefore the main work can be seen as observation and checkup not creation and formation. While several known desktop CAD applications like autoCAD have introduced their smartphone version, several other new applications are present such as Infinite Design, cadTouch, A360 and eDrawing. Figure 2.3 [77] shows a developed CAD application for medical sensors and devices.



Figure 2.3: Medical CAD Application [77]

In parallel, Computer Aided Manufacturing (CAM) can be seen as the use of computer-based systems in order to manage, plan and optimize the operations of a manufacturing plant through a given computer interface, direct or indirect with the production resources of the plant. The interest in CAM grew because of the advantages that accompany it in developing the capabilities of the company through the combination of traditional economy modes of operations with methods and means of flexibility and efficiency. The main difference between CAM and other applications listed so far is the fact that they connect the user to hardware equipment whether it is a manufacturing machine or a motor or gear whereas the previously cited applications offer a virtual output.

Seeing that CAM computer applications are slowly becoming a staple in modern manufacturing, an interest in such implementation on smartphone is obviously seen even at a much smaller scale. A prime example of such is the Arduino mobile application that can connect the user to an Arduino Printed Circuit Board (PCB) via their smartphone through a WAN connection. However the Arduino PCB and the connected mobile applications are of educational and personal usage purposes such as for students and enthusiasts and not for heavy machinery and high stake professional use. An extension of such use is the implementation of a house monitoring and control system that enables the user to control various features of an environments like lighting, heating, security through the smartphone platform. Known manufacturing applications on the smartphone platform include Manufacturing 360, LeanApp and Manufacturing MRP.

2.2.5 Data Collection of Physical Phenomena

Smartphones have already built-in sensors that can be used for scientific measurements where they can be considered a multi-functional tool such as in magnetometry for detecting safe electromagnetic devices [78]. Other uses include tacheometry and telemetry. An example of this is the work of Dong-Hoon et al. [79] which aims to build an application that aids in the identification of a Volatile Organic Compound. This is through the collection of the standard testing paper result through the cellphone camera and the identification of the gas by analyzing the visual results with the application's database. Another example of this is the work of Angela et al. [80] which introduces a smartphone-based instructional particle image velocimetry educational tool for Android that is open-source and low cost. The concerned work offers users the guided instruction that is backed by the technological infrastructure and allows them to visualize and study authentic flow fields in real time.

In addition to data measurement, smartphones can be used as widespread pinpoint devices for conducting wide surveys and mass data collection because of their large popularity and ease of carry. An example of such usage is in meteorology where a smartphone network can be implemented to monitor tropospheric delays and variations based on the changes in data collected by smartphone sensors [81].

2.2.6 Discussion

The examples and work shown above vary between several domains and fields. However, they share several similarities. For instance, a large portion of them heavily relies on the use of one or several of the smartphone's sensors and other hardware features mainly the camera. The related components are the central point of operations, and the software/simulation to accompany it is secondary or just an attachment to the main operation. The work shown highlights the smartphone as a primary tool for measurement with less focus on computing and simulation. In addition, the presented state-of-art might not completely encompass all of the recorded and published work, but most of which has been brought to light in recent years. Therefore it is apparent that there is a certain lack in the density of the work surrounding the topic despite the surge of interest. This relatively small amount can be contributed to several factors such as the increase in investment in cloud computing, rapid development of smartphone capabilities or other.

Thus, as stated before, scientific smartphone applications are mainly being developed with the focus on smartphones as being a compact set of tools all in one device, or as a visual enhancement to the camera's input as it is seen with augmented reality, or as portals for desktop work such as blue prints and designs. However, work related to utilizing the smartphone's computing capacities for simulation is still small and young if not non-existent. With the lack of proper direct work towards smartphone scientific applications for simulation, any work that is related is seen through the introduction of new algorithms that replace old ones with same efficiency and output. This is accompanied with less computing and memory requirements such as the introduction of less-demanding iterative methods for standard used direct methods that are more demanding.

It is to be noted that the reliance on the hardware development of smartphones to be able to accommodate for more demanding simulation applications might not be feasible because desktop applications have shown to be advancing in requirements at a faster capacity. While smartphones are less likely to fully replace desktops in use and purpose, a large portion of scientific concepts and methods can be developed to remove the need of desktop for support.

2.3 Applications Development Challenges

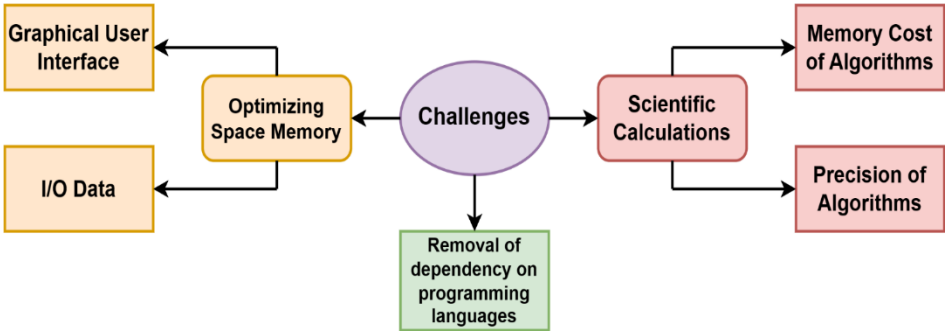


Figure 2.4: Smartphone Scientific Application Challenges

In this section, the main challenges for the development of scientific smartphone applications are identified alongside a prominent literature solution for each. While other challenges are present and can influence the success of the development of the smartphone scientific calculation, the following have been found as the main ones for implementation and development. Figure 2.4 shows the challenges in discussion. They are primarily the result of the smartphone's limited memory capacity.

2.3.1 Human / Application Interface

This section focuses on all computational challenges that primarily result from data display through the Graphical Interface as well as the management and transition of data from the user and to him.

2.3.1.1 User Graphical Interface

Graphical User Interfaces (UI) in scientific applications tend to be heavy in performance demands because of the standard amount of information and data that needs to be displayed and manageable by the user. Scientific applications mostly have a unique and dedicated interface because of the difference between their respective fields.

Concerning mobile applications in general, the scientific study of usability guidelines for mobile applications is still young in development. Most of the conventional guidelines are based on common attributes between established mobile applications [82]. Despite the risks of increasing memory costs, UI studies and developments are generally designed with user preferences and tendencies as the highest priority [83]. An efficient UI offers the basis for improving the overall performance of the system [84]. A proper balance must be found between user satisfaction and memory costs where the latter must not heavily compromise the first. Several high quality applications in the market have not gathered sufficient popularity due to complexity and unattractive or confusing UI [85].

One of the current prominent and constantly developed methods for User Graphical Optimization is Combinatorial Optimization [86]. Its main potential lies in its capability of searching for large number of designs which complements human designers' work. The process of Combinatorial Optimization is shown in Figure 2.5.

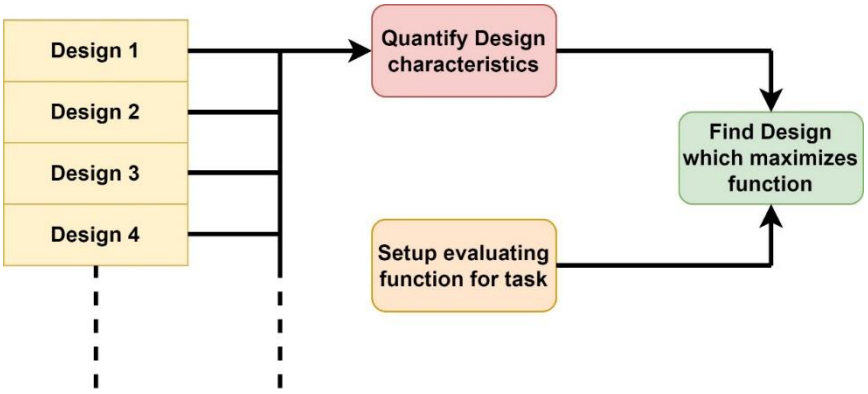


Figure 2.5: Combinatorial Optimization Process

The design of the GUI revolves around the algorithmic process of combining design decisions in order to obtain acceptable or optimal solutions which are defined as an objective function. Afterwards, the design is defined by the process of trying several combinations of various decisions that offer the highest value of the defined function.

The main applications for Combinatorial optimization are in generative design where the aim is to solve problems with a large number of objectives and to algorithmically construct interfaces with high quality for the end users [87]. Another application is in interactive design where optimizers may aid in completing partially accomplished work and assessing them and in turn defining a rationale for them.

In addition, combinatorial optimization is employed for adaptation and personalization where its objective function can represent individual-specific or moment-by-moment requirements. This is seen in the re-optimization of the design when the orientation of the device is changed [88].

While Combinatorial optimization is constantly seeing an increase in development and related work, its main disadvantage lies in its need for an accurate approximation of computational efficiency of functions and displays before design generation, which can greatly vary in an engineering application depending on chosen parameter size and simulation time. In addition, state-of-art models have been found too inefficient for large optimization problems where a specific approach is required for each domain and application design [89].

2.3.1.2 Data Management for Input/Output

Scientific applications tend to rely on a large set of input and output data for simulation and execution. Deelman et al. [90] show the difference in data management demands between standard and scientific applications because of the data-intensive nature of scientific workflow in these applications.

While this cost can be negligible on desktop platforms because of the standard large space available, it becomes far more noticeable on smartphone platforms [91]. This is also impactful because of the large amount of memory and disk space occupied by built-in systems and applications [92]. The optimization of saved data does not only improve disk space costs but

the efficiency of access to this data [93]. In addition, the work of Carroll et al. [94] highlight the direct relation between storage size and energy consumption on smartphone platforms.

One of the emerging systems for management and storage of data is Cloud Computing [9]. It is the delivery of computing services which include servers, storage, databases, networking, software, analytics and intelligence over the internet. In addition to managing storage and transaction of I/O data, cloud computing can also be used for handling operations and simulations server-side.

While several works have defined varied versions of Cloud Computing Architecture, the system is mainly divided into the management and the core stack. In turn, the core stack is divided into the resource, platform and application layers as seen in Figure 2.6. The resource layer is the infrastructure that is composed of both physical and virtual computing and handles the storage of data. The platform layer is divided into several sub-categories which include the computing framework which manages the transaction dispatching and task scheduling as well as a storage sub-layer which provides caching and storage capability.

The management of I/O data is divided between these three layers where the resource layers manages the storage of data, the platform layers manages its transaction and the application layer handles the presentation of data and the interaction with the user.

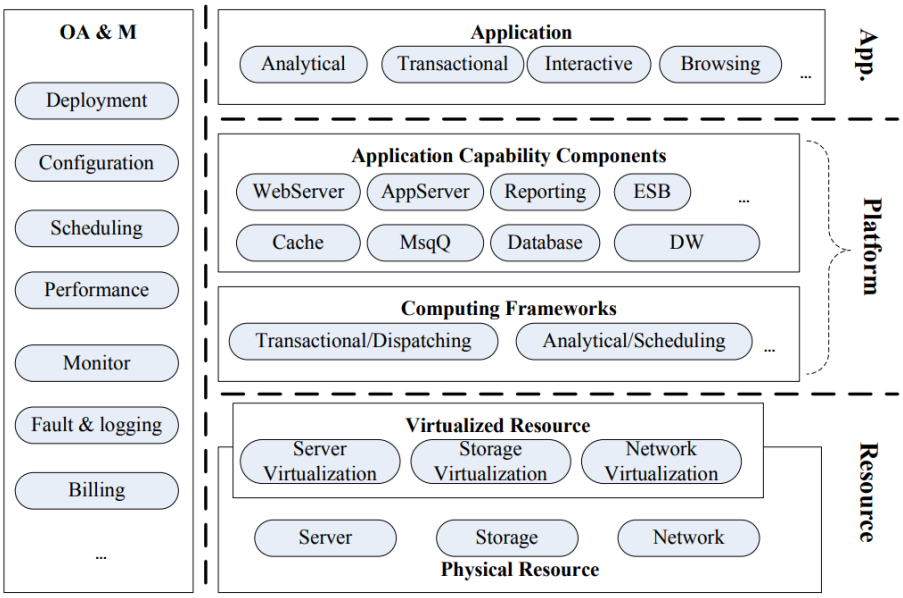


Figure 2.6: Cloud Computing Architecture [9]

However, one of the main drawbacks of Cloud-computing is that it requires a constant internet connection in order to establish a server connection. This requirement also leads to other challenges and risks that are not present in local and offline applications which is data breach, privacy risks and the need for robust security [95]. Moreover, the required server connection and the constant transaction of data between the targeted platform and the concerned server leads to a serious increase in energy consumption which impacts smartphone battery and hardware durability [96].

Another emergent technology is the Burst Buffer System which was created in order to adapt to the rapidly rising data pressure from scientific applications. It is considered by many to be a promising solution to the growing I/O challenge on High Processing Computer system. The primary strategy is to leverage an external storage system with several tiers of high-speed storage devices between computing nodes [10]. Despite these benefits, most work currently focus on simulation and modeling of burst buffer where design and implementation are rarely documented [97]. Figure 2.7 shows a standard burst-buffer system architecture [98]. In short, it is a fast intermediate storage layer positioned between the front-end computing processes and the back-end storage systems. It bridges the performance gap between the processing speed of the compute nodes and the I/O bandwidth of the storage systems. Burst buffers are often built from arrays of high-performance storage devices, such as NVRAM and SSD. It typically offers one to two orders of magnitude higher I/O bandwidth than the back-end storage systems.

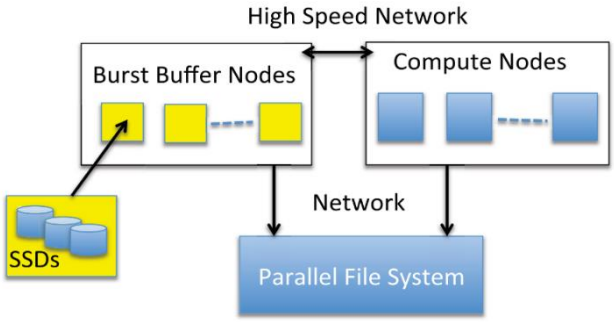


Figure 2.7: Standard Burst-Buffer System Architecture [98]

While Burst-buffer systems are steadily gaining in promise and potential, its main disadvantages lies in its core design that targets high-end computers which in turn leads to poor implementation on smartphone [99].

2.3.2 Algorithm Execution

This section prioritizes the computational complexity related to the execution of algorithms after their reception and setup. The challenges described here are independent from the visual representation of the data as well as anything related to storage and data management.

2.3.2.1 Memory Cost of Algorithms

Scientific algorithms can lead to a large increase in complexity and memory costs for execution. Complexity results from the large and increasing scale of operations and simulations. A common example of this is matrix inversion. Ataeeshojai et al. [100] and Albreem [101] show the challenges in matrix inversion despite modern computing. The cited work addresses the complexity issue on desktops which would be more staggering on smartphones. An algorithm's performance varies based on external and internal factors. Memory space requirements, as well as computation time, are 2 of the main influencing factors in the implementation of computational algorithms. External factors usually vary between platforms and software whereas internal factors are related to the efficiency of algorithms both in memory space usage and time [102].

While its values are usually taken in certain consideration when it comes to desktop platforms, it is far more important in regards to smartphones because of the limited hardware/software constraints and memory constraints. An algorithm that is considered feasible on desktops based on Big O value can simply be non-executable on smartphones because of the high difference in memory capacity. It is to be noted that the desired feasible memory cost would always be the smallest possible because of the nature of smartphone memory consumption that is usually split among several background applications and mobile services.

One of the main developed literature technology in this domain is memory parallelism which allows simultaneously executing more than one operation across several processors within a common node as highlighted in Figure 2.8. Another developed iteration is the Distributed Memory Parallelism where an application runs multiple instances of itself across multiple nodes. In both methods, the processors are connected via a message passing interface.

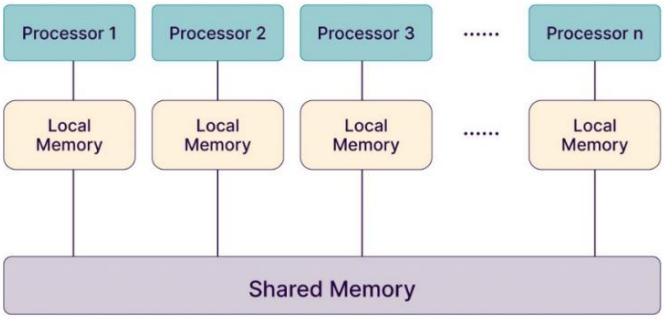


Figure 2.8: Shared Memory Parallelism

While Shared Memory Parallelism is a standard concept in most modern CPU multi-core processors, the concept is also seeing interest on the smartphone platform where instead of just splitting operations across the cores of the smartphone CPU processor, the operations are also split upon several separate smartphone platforms after establishing a connection. An example of this is the Distributed Programming Layer over Mobile Agents (DIPLOMA) [103], a programming layer and distributed shared memory system that provides coherent relaxed-consistency access to data residing on different mobile phones across a large geographic area.

The main disadvantage of establishing Shared Memory Parallelism across several separate smartphone devices is in its core design and that is the need for several platforms in order to achieve this which is a serious requirement that removes user independency as well as introduces the risk of data breach.

2.3.2.2 Precision of Algorithms

A large portion of the standard algorithms used in scientific calculations are approximation algorithms where the output has a provable guarantee on the distance of the returned solution to the optimal one [8]. These are commonly used in optimization problems that cannot be solved exactly in polynomial time. While iterative algorithms optimize memory costs, their implementation can result in increased runtime. This increase is due to the linear execution of operations [7].

In addition, standard iterative algorithms can have varying output efficiency based on the scenario and environment of execution. A common example of this in algorithm execution is the Ordinary Least-Squares Method which is a core process in Control as it is one of the main operations in Model Identification.

2.3.2.3 Removal of Dependency on Programming Languages

Programming language compilers are costly on memory because of the compiler's limited knowledge of the runtime execution of the compiled program [104]. Non-optimized written codes can remove the consistency guarantees of the program both on memory cost and runtime [105]. In addition, one of the major requirements of programming languages is skill, experience and a deep knowledge of the meta-models involved [106]. This can directly impact memory costs because of the risk of non-optimized written codes. Therefore dedicated programming languages on smartphone are not common on the platform because of the high memory costs as well as the resulting disadvantages.

For such reasons, the challenge is not to adapt the same concept of programming languages but to simply provide its advantages which are mainly the customization possibilities they provide. In addition, this should also account to the scalability potential that is offered with the use of programming languages, therefore allowing the customization of both the details and size of the targeted projects.

The evolution in the domain has been first sighted by the transition from text-based language towards the implementation of Model Transformation Language (MTL) [107]. Model transformations are considered the basis of Model-Driven Engineering (MDE), and dedicated MTLs emerged with the popularity of the MDE paradigm. MTLs claim to increase the ease of development of model transformations by abstracting from recurring transformation aspects and hiding complex semantics behind a simple and intuitive syntax. Nonetheless, MTLs are rarely adopted in practice, there is still no empirical evidence for the claim of easier development [108].

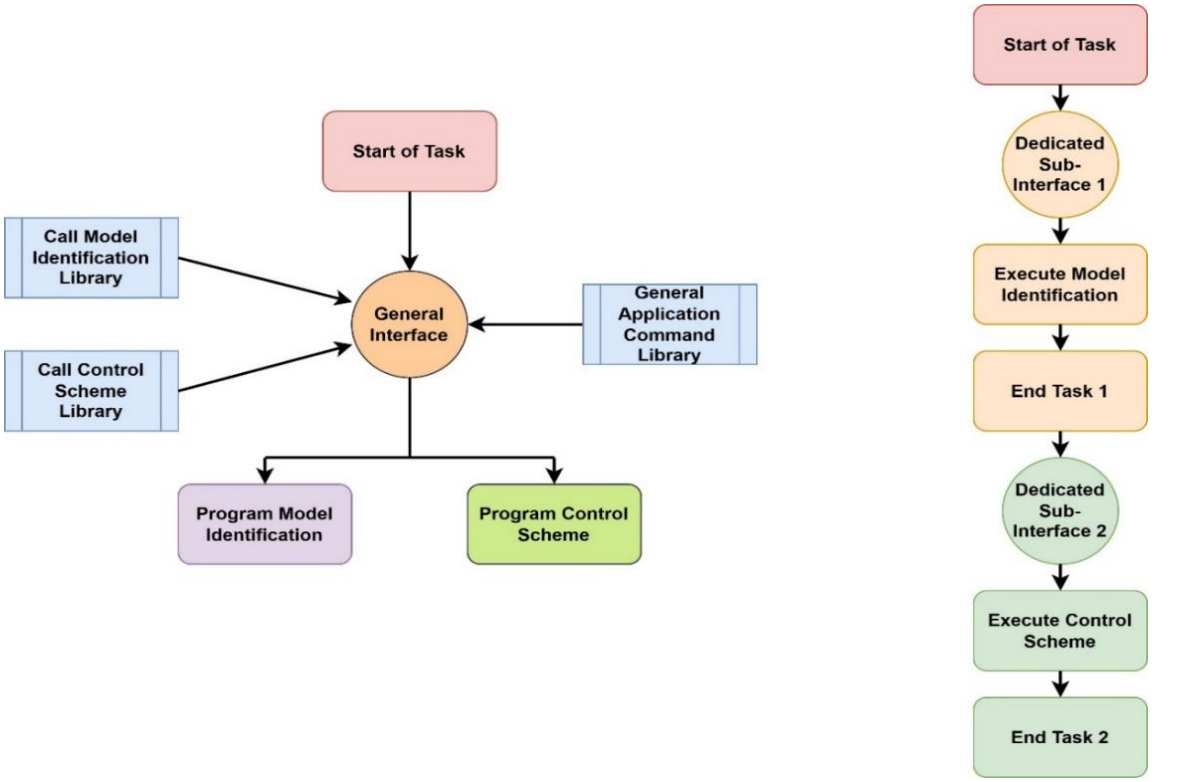
2.4 Proposed Software Solutions

In this section, a set of solutions is proposed for the previously discussed challenges. These solutions are based on software optimization and were compared with the implemented methods in other standard scientific applications.

2.4.1 User Graphical Interface

This solution is based on the design of the used toolbox that enables the implementation of a varied set of dedicated sub-interfaces that spreads the tasks between sequentially connected and separated interfaces. In addition, the step-by-step filtering of the data to be used and methods reduces the memory costs behind simultaneously displaying and utilizing all of the resources needed. From the application's side, this filtering can be applied not just on the possible options but also on the mathematical method choice as well. Moreover, the sequential filtering would allow the application to execute smaller operations between each step. This decreases the number of operations conducted in the final execution therefore splitting memory

cost over the entirety of the process and not conducting all operations in one execution. This solution can be further explained by examining a task that is handled in a conventional interface and then in the proposed one. The concerned task is to identify a system through Model Identification and then design and execute the control process. The difference between the two interfaces is shown in Figure 2.9.



(a) Conventional UI Design (b) Proposed UI Design

Figure 2.9: UI Design Comparison

In the Conventional Interface, all of the libraries and tools for the two tasks are present. The user cycles between the given options and commands in order to design the execution of the two given tasks. The output of Task One and that of Task 2 are simultaneously displayed for the user therefore increasing after task-execution memory cost as well.

In the Proposed Interface, the tasks are distributed on two dedicates sub-interfaces. Only the libraries used by the task are called and presented in the concerned sub-interface alongside the tools. After the execution of Task One, the progress is saved to be used in the second sub-interface after the memory cost of the first sub-interface is cleared. Figure 2.10 details the memory costs both before Task One execution and after its execution and the same data for Task Two and shows that the maximum after-execution memory cost of the 2 tasks is 11MB.

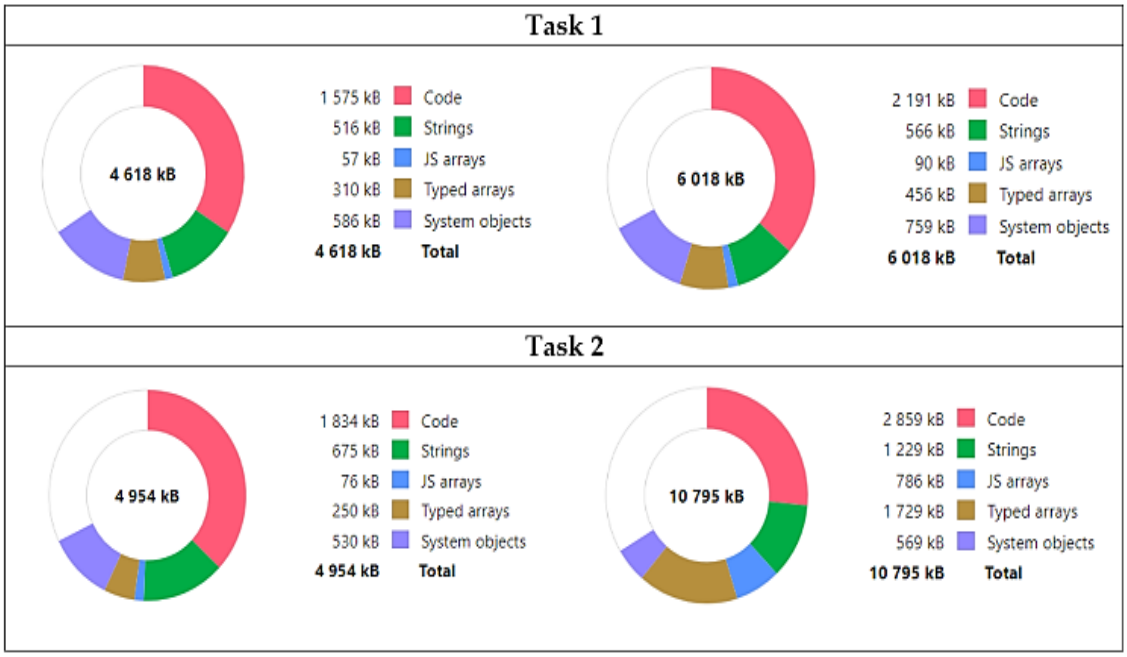


Figure 2.10: Task 1, 2 Memory Cost Pre-Execution and After-Execution

This is in comparison to the minimum memory requirements of other standard known scientific applications for control as seen in the table below and after taking into consideration possible future scaling and worst-case-scenarios for the costs of the proposed solution. Table 2.1 shows the minimum set requirements for these standard applications.

Application	Minimum Memory Requirement
Matlab	8 GB
Labview	4 GB
Wolfram Mathematica	4 GB
HSC	2 GB
P-Spice	8 GB
Proposed Solution	Max 100 MB

Table 2.1: Minimum Memory Requirement for Standard Engineering Applications

2.4.2 Data Management of Input/Output

The primary approach for this solution is the design of a model that ensures Space Memory Optimization and allows integration of operations across other software. Even if full integration with simultaneous running of operations across several software is not yet achieved, the first aim is to offer a model that allows the extraction of data at ease to be used in other software.

In order to ensure this, the saved files are in a .txt format. They are a form of a 1d array that contains all of the information needed from an operation. This is based on the fact that each presented scenario and mode of operation for the user has a built-in progress save format. This is achievable because of the built-in control schemes offered for the user as well as the design of a separate model identification process. A template is created for each of the specific sub-interface tasks and the user only fills the parameters which are transitioned to the template

saves. For example, in the case of identification of a given system or sensor, the template for the identified model is created via a 1d array that stores all data in a specific intended order. A template example is shown in Figure 2.11.

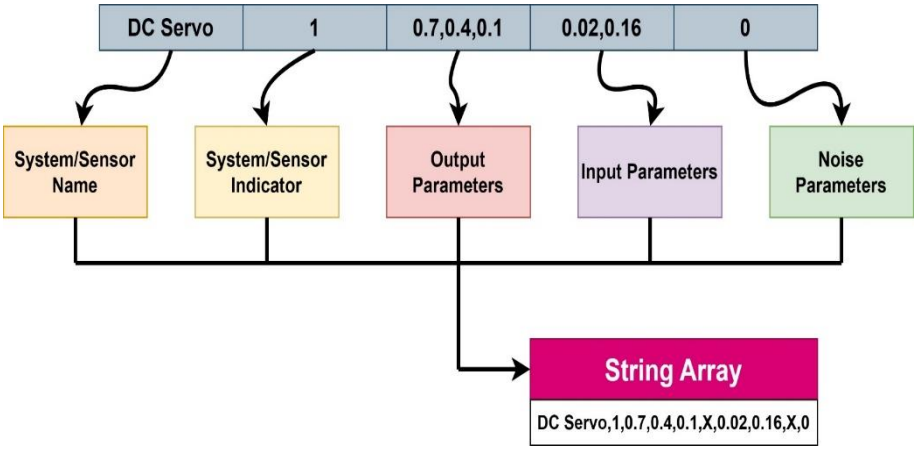


Figure 2.11: Identified Model Save Template

The Data is stored sequentially in a way that is identifiable by Load operations without any need for a type format that is specific to the application. This would also allow the extraction of this data even outside from the application and be directly read from the Save to another software. This concept is also applicable to a control scheme save. Figure 2.12 highlights this difference between a conventional and the proposed save template for a control scheme.

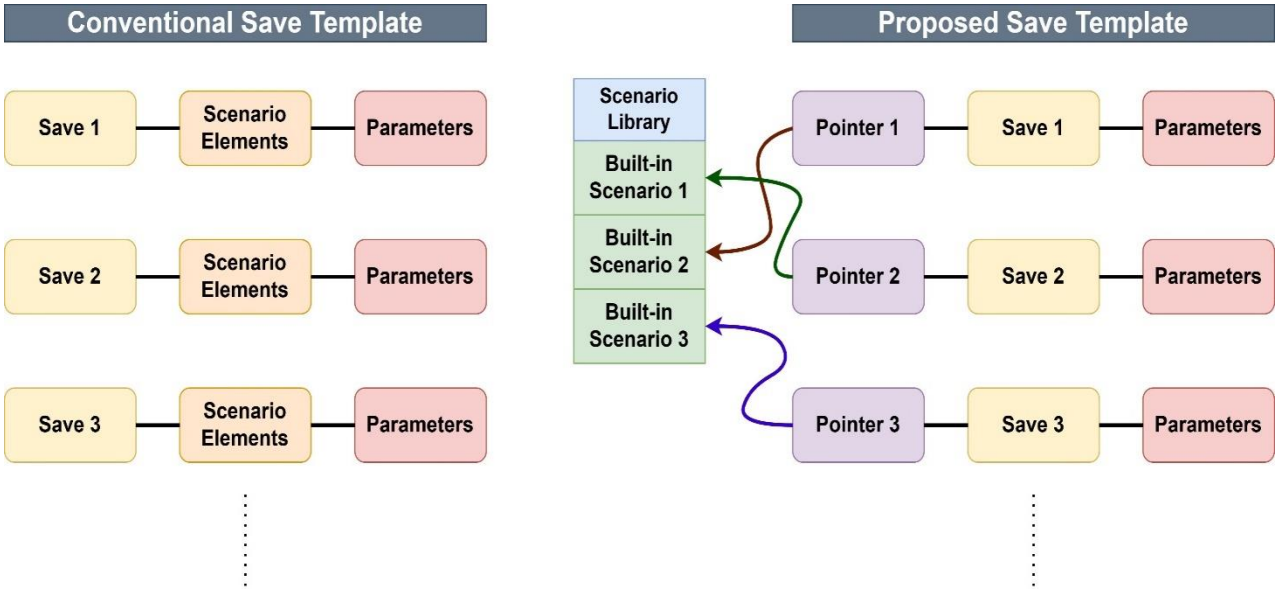


Figure 2.12: Conventional Save and Proposed Save Template

The conventional save indicates the save template of standard control software such as Matlab. All control blocks are first custom built and placed by the user. This would also include saving the data related to control blocks specifications such as block size, x coordinates, y coordinates, links and other. Other than this, there are also external components that can be added to a Save such as images, external code or other.

In the Proposed Save Template, all control blocks are already built-in and the user chooses the one that suits his work. The added pointer is to track which control scheme is chosen. Figure 2.13 shows the presented set for the user to choose from the smartphone application.

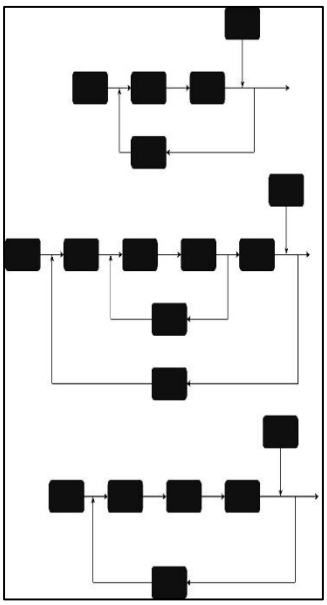


Figure 2.13: Control Scheme Choice

Figure 2.13 shows the set of built-in control scheme templates that are available for the user to use for building his design.

The pointer not only indicates which control block is chosen but also the data and data sequence in the same template. This also removes the need for any decoding that is specific for the application in order to be able to retrieve the data.

In order to show the difference and impact between the Conventional Save Template and the Proposed Save Template, a comparison between the proposed template for the application and the save template of Matlab is presented. The comparison is based on already built-in examples in Matlab and built-in control example in the proposed Control application. Figure 2.14 shows the storage used by Matlab Examples.

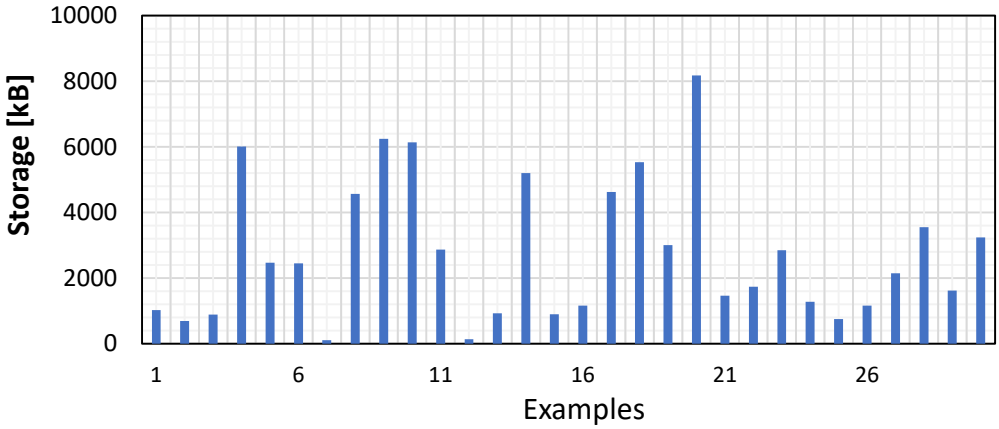


Figure 2.14: Matlab Built-in Examples Memory storage

Figure 2.14 shows the relatively large required storage by most Matlab Examples where the average of the data collected is 3MB. Note that this does not include large scale built-in example projects in Matlab such as Antenna Example or Bluetooth. The second sample is based on the built-in control examples that are given in the proposed application. Figure 2.15 shows the sample memory sizes.

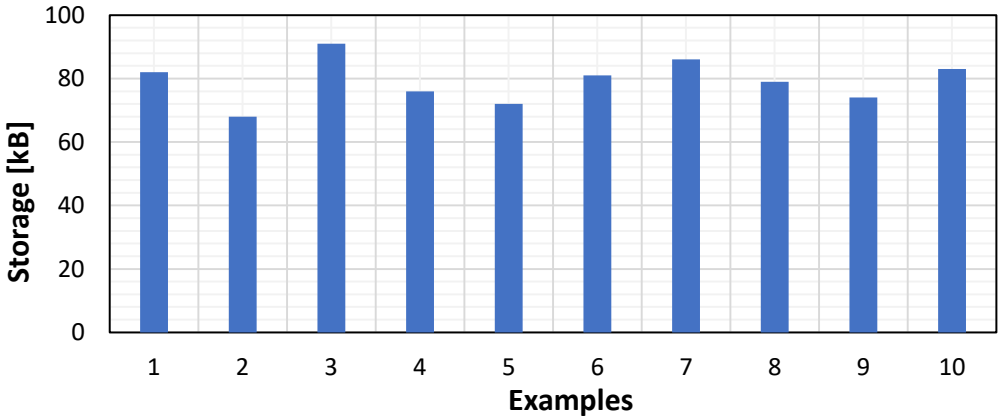


Figure 2.15: Proposed Application Built-in Examples Memory storage

As evident between Figure 2.14 and 2.15, the proposed built-in templates offer a much smaller storage memory requirement.

2.4.3 Scientific Calculations

This challenge is primarily addressed in the second chapter of this work where the focus is on mathematical optimization of algorithms. However, software optimization can also aid here which is through design choice. The proposed solution would be the implementation of a dedicated toolbox that offers the user all of the standard and unique but feasible options in the domain of control. The toolbox must cover these options while not overwhelming the user with all of the options simultaneously. This would be a systematic approach that would sequentially filter the user's options based on their step-by-step input and choices. This methodology is not present in the currently used languages because they technically permit the user from compiling a code that sequences a set of options that are not in principal connected nor valid in the same situation or system. Figure 2.16 shows this filtering toolbox in the control process of the application.

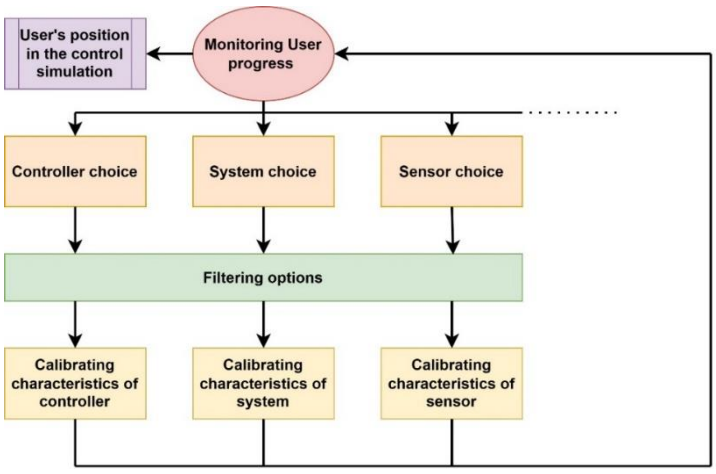


Figure 2.16: Control Process Toolbox

From the application's side, this filtering can be applied not just on the possible options but also on the mathematical method choice as well. It is to be noted that the choice of mathematical method would be irrelevant for the user if the possible methods in question would return the exact same output which is the case in the data search example. Therefore the toolbox's focus on delivering to the user all of the aspects that might alter the output would remove all the complications and increase in operations that might result from syntax or other details while preserving the customization options.

In addition, the advantages of toolbox implementation are amplified by the smartphone platform. A toolbox would remove the inefficiencies in execution through compiling a programming code and synchronizes well with the standard smartphone application's general orientation of providing a simple result rapidly and easily. Moreover, the sequential filtering would allow the application to execute smaller operations between each step. This decreases the number of operations conducted in the final execution therefore splitting memory cost over the entirety of the process and not conducting all operations in one execution.

2.4.4 Dependency on Programming Languages

The proposed solution must remove the disadvantages while safeguarding the main advantages of customization. The objective would be to provide an approach that offers customization and accessibility without programming skill requirement while satisfying smartphone constraints. Figure 2.17 shows the process of control from Model Identification to Control Process in the control simulation application prototype.

The main task is to identify a model based on the data of an observed system and to design and test the control process of the obtained model. During the first step of model identification, the user is provided with a list of options for the method alongside the type of the resulting model. Each option is followed by a list of its parameters as well such as number of input parameters and output parameters for the chosen model type. Afterwards, the user saves the system and moves on to the control process. The same customization potential is presented. The user first starts by choosing the general design of the control process which is based on a broad range of standardized and conventional designs. The user then proceeds to customize the parameters of each block of the control scheme. Same as before, each block option chosen is followed by a subset of options and parameters to choose and tune. An example of this is the controller type choice; a chosen PID type controller presents the user with the Kp,Ki,Kd parameters. Optional components are provided as well. An example of this is the ability of adding noise to the system whether it is Gaussian, Poisson or other or to not have any type of noise. While the applications aims to provide all of the possible combinations and design options, this task is considerably staggering in systems control because of the large set of possible scenarios. For such reasons, the success of this approach relies on the knowledge and capabilities of the developer in the concerned domain.

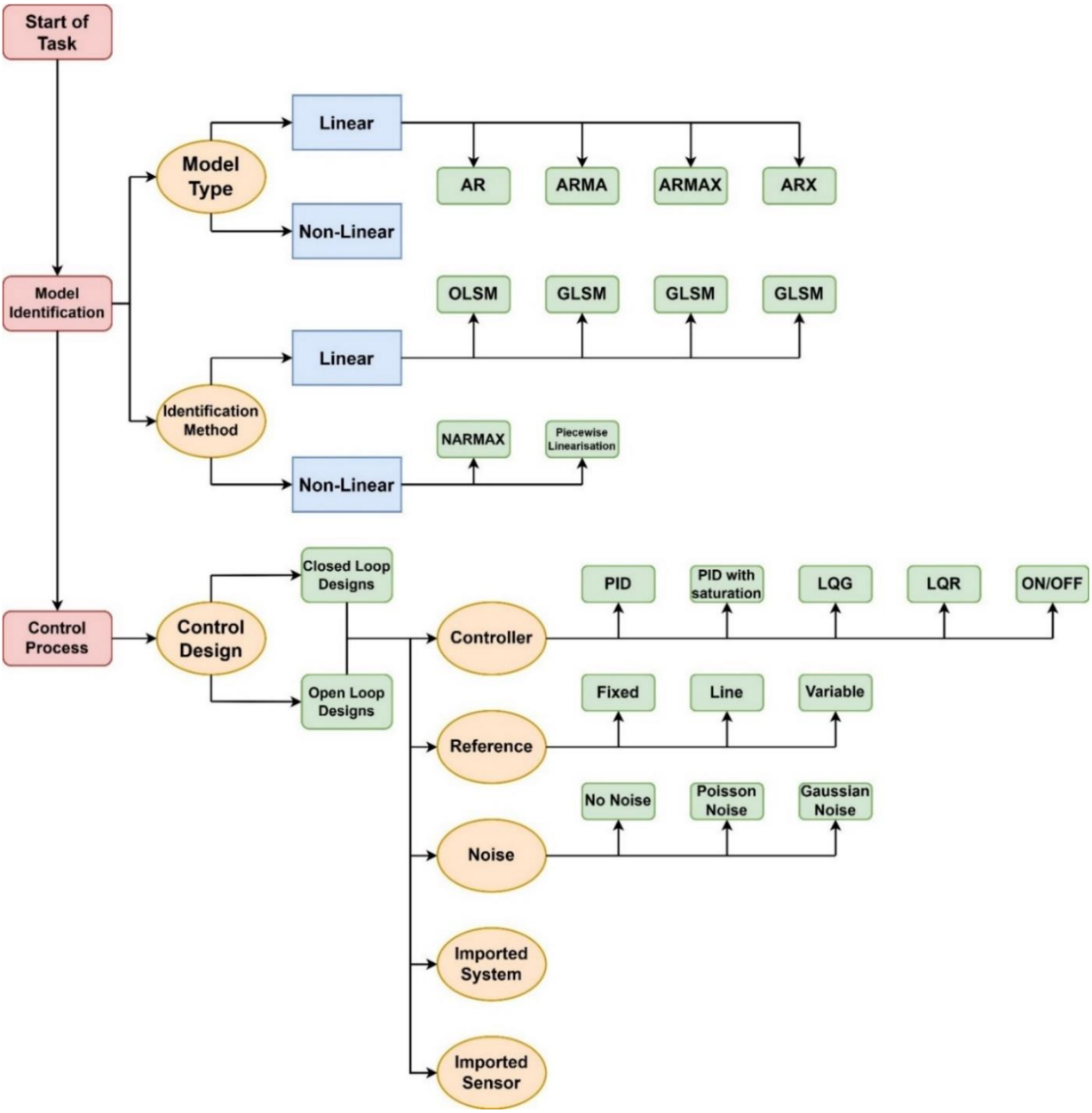


Figure 2.17: Control Process Task Options

2.5 Conclusion

The work presented in this chapter offer a set of guidelines for the development of engineering applications on the smartphone platform, both through the defined challenges that offer a set of predictability for possible difficulties and the solutions who present broad guidelines for development. However, a large segment of problems in the domain of Control are standardly known for their computational complexity and need for high resource allocation. The next part of the work focuses further on some of these specific problems where the approach follows a mathematical optimization to target the operations that formulate these problems.

3 System Modelling: Identification Algorithm

3.1 Introduction

The following chapter focuses on the introduction of a new identification algorithm of a linear system specifically for constraint simulation. This revolves around the identification of the underlying parameters that define a specific system depending on its input data and output data. Considering the linear system as shown in Figure 1.2 with form

$$Y = AX \tag{3.1}$$

where $X \in \mathbb{R}^{n\times m}$ denotes the input, $Y \in \mathbb{R}^{1\times m}$ represents the output, $A \in \mathbb{R}^{1\times n}$ is an unknown matrix, *n* and *m* are the system's number of inputs and number of observations respectively.

The goal of identification algorithms for the system is to identify the matrix A based on the output matrix Y and input matrix X. While the primary focus of most of the standard identification algorithms was to increase the precision of the model, their implementation and operation can lead to large memory costs and high runtime which leads to a deterioration in operations on weaker platforms such as smartphones [91]. This is also noticeable because of the large amount of memory and disk space occupied by built-in systems and applications [92]. In addition, the effectiveness of literature identification algorithms is dependent on the system in study as well as the overall goal of the operations [109]. An example of this is seen in the derivatives of OLS which include Partial-Least-Squares (PLS) and Regularized-Least-Squares (RLS) with each having varying performance results based on the given scenario. PLS run time is slower than other methods and is only efficient when variable numbers are more than compound numbers in the data set [110] and RLS is more efficient only when the coefficients are penalized or else it has more run time for same efficiency [111]. These methods are more efficient than others in certain scenarios and when certain criteria are met. A general approach is GLS but it shows a general difficulty of implementation which leads to inefficient execution, long run time and the need for trial and error [112]. In the case of the proposed algorithm in this work, the main objective is to reduce complexity while maintaining or increasing output precision. The suggested algorithm would therefore aim to reduce memory/time costs in order to account to the hardware/software costs.

This chapter is organized into four sections. Section 3.2 concerns the challenges and the concept of the proposed algorithm. The comparison between the different proposed versions of the algorithm is in Section 3.3. Section 3.4 shows the comparison with chosen literature methods for identification and Section 3.5 concerns the developed generalized form of the proposed algorithm.

3.2 Proposed Algorithm Design

This section presents the challenges related to the identification process where Figure 3.1 offers a summary of the discussed challenges here. Afterwards the principle of the proposed process is set which leads to the statement of the algorithm.

3.2.1 Challenges for Identification

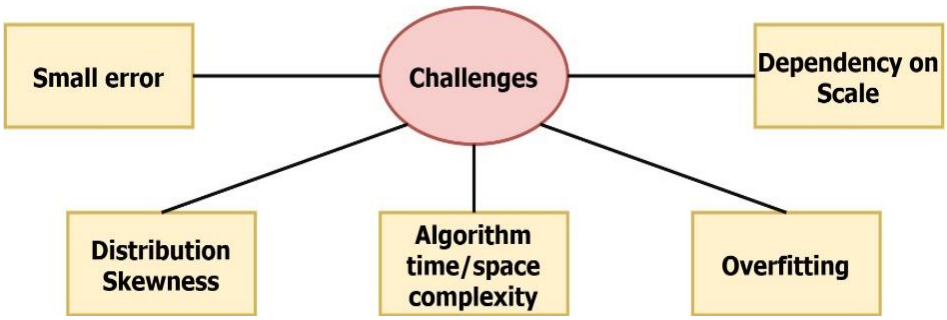


Figure 3.1: Challenges for Identification Process

The source of inspiration for the design of this algorithm is the challenges highlighted by other identification algorithms methods. In order to explain them, the relative error is defined as

$$\mathfrak{e}(i) = \left|\frac{Y(i) - Y^p(i)}{\mathfrak{v}(Y(i), Y^P(i))}\right| \quad \text{where } i \in [\![1, N]\!] \tag{3.2}$$

where $|.|$ is the absolute value of a real number, $[\![.,.]\!]$ denotes the set of integers in a range, N is the sample size used for the identification process, $Y^p(i)$ is the predicted value of $Y(i)$ at i and the function $\mathfrak{v}$ is defined by

$$\mathfrak{v}(x, y) = \begin{cases} x & \text{for } x \neq 0 \\ y & \text{for } x = 0 \text{ and } y \neq 0 \\ 1 & \text{for } x = 0 \text{ and } y = 0 \end{cases}$$

Note that $\mathfrak{v}(x, y)$ is used to guarantee a non-zero denominator in the observed error.

In addition, the residual error is also used for comparison in this section and is defined as

$$\mathfrak{s}(i) = |Y(i) - Y^p(i)| \tag{3.3}$$

Moreover, for the sake of examples here, the I/O data of the Centrifugal Pump is used here. This system is used to move fluids through the conversion of kinetic rotational energy into hydrodynamic energy. It is composed of a volute casing and an impeller were the fluid is moved via centrifugal force. The input of the system is the flow rate and the observed output is the generate pressure by the pump. The Centrifugal Pump hardware parameters are given in [113] alongside the model equations. Figure 3.2 shows the concept of the system and Figure 3.3 the corresponding I/O Data.

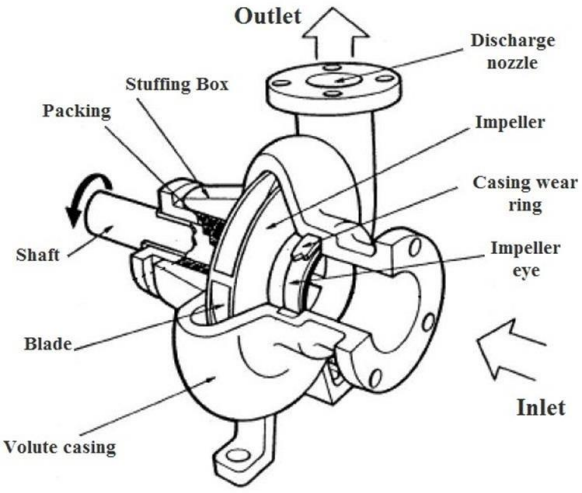


Figure 3.2: Centrifugal Pump Concept

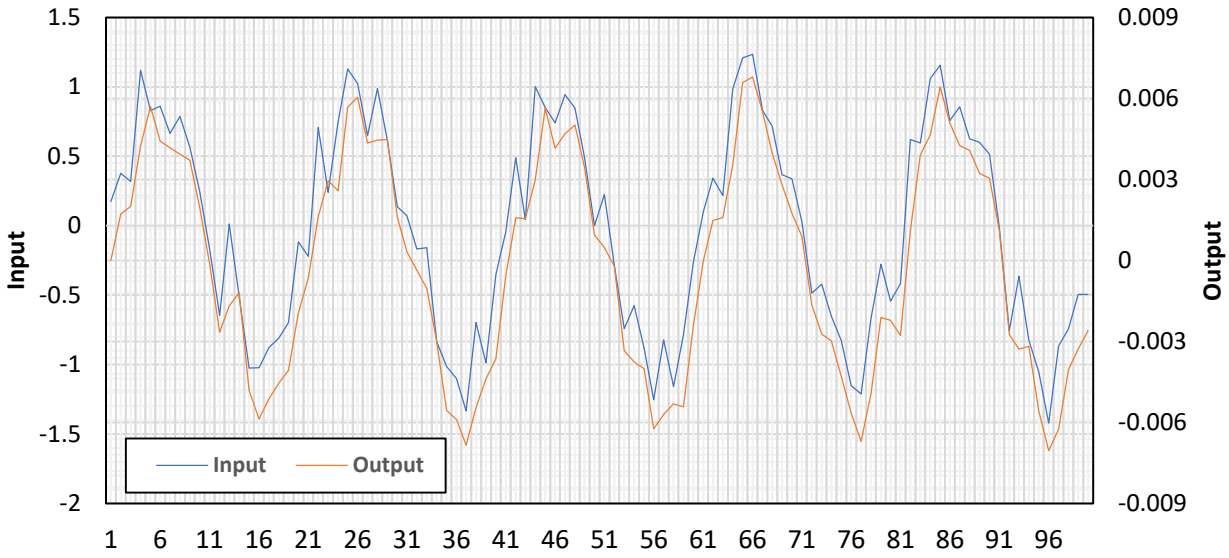


Figure 3.3: Centrifugal Pump I/O Data

3.2.1.1 Small Error and Heteroskedasticity

While Ordinary Least-Squares is the basis of standard FGLS methods, OLS residual differences tend to be minimal and might not fully represent the variation in observation of a given sample. The work of White [36] was first developed to address the issue of heteroskedasticity. In turn, the methods introduced later on [37] aimed to counter the issue of small OLS residual difference. The main advancement in the proposed method by Horn et al. [114] is that it is unbiased in the case of homoscedasticity of the errors.

However, in [37], it is shown that in the case of homoscedasticity, the OLS method is still considered more reliable than the proposed FGLS methods. The tests conducted show that there isn't a method that is the most efficient in all scenarios and the chosen method is dependent on the given system.

3.2.1.2 Distribution Skewness

In a mean regression-based method, the mean is sensitive to outliers and may perform poorly when the distributions are skewed [115]. For these reasons, works such that of Yuan et al. [115] introduce median regression analysis which focuses on the relation between the median of dependent variables and the independent variables. For these reasons, a differentiating index position, such as the median in [115], can be introduced normalized the error distribution.

3.2.1.3 Algorithm Time/Space Complexity

As seen previously, scientific algorithms may lead to a large increase in memory costs. In the case of identification algorithm, this complexity can result from the large and increasing scale of operations and simulations.

In the case of obtaining $\boldsymbol{\Omega}$ for (1.3), the standard literature methods rely on several operations of matrix inversion and multiplication. They also rely on several iterations until the desired threshold of precision is achieved. These methods are further explained in Section 3.4.

Considering (1.3) where, for example, $A \in \mathbb{R}^{1\times 5}$ and $Y \in \mathbb{R}^{1\times n}$, the resulting number of operations is found as

$$O\big((1.3)\big) = \frac{9}{2}\, n^3 + 4\, n^2$$

Figure 3.4 shows the variation of O ((1.3)) in function of n. Note that (1.3) is only a part of the calculations needed for the FGLS method.

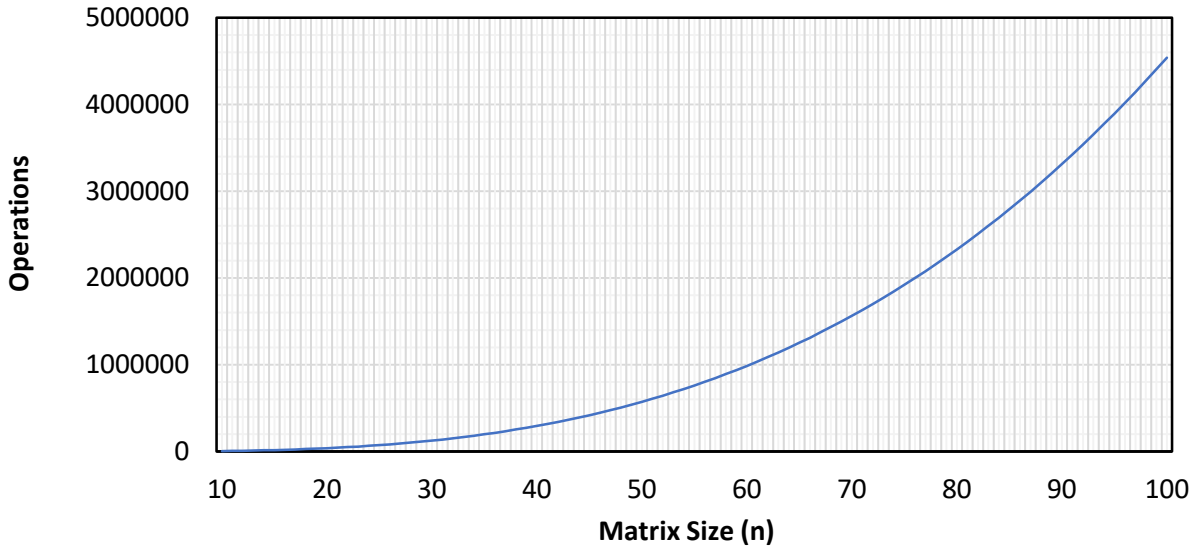


Figure 3.4: Variation of FGLS number of operations

Figure 3.4 highlights the rapid increase in the number of operations needed for the execution of the algorithm and in turning resulting in larger computational costs.

3.2.1.4 Dependency on Scale and Overfitting

Small residual differences can also lead to overfitting[9] and difference in model performance on varied data ranges. Several works [116] [117] employ regularization techniques to prevent overfitting. Some work [118] focused on the use of relative error instead of residual error to lower the risk of overfitting by removing the dependency on the scale of the used dataset for observation. In addition, the use of relative error allows for further interpretation of the performance of the algorithm. This is highlighted in the work of Kat et al. [119] which employs the relative error. Figure 3.5 shows the residual error from the identification process of Centrifugal Pump I/O Data and Figure 3.6 shows the relative error.

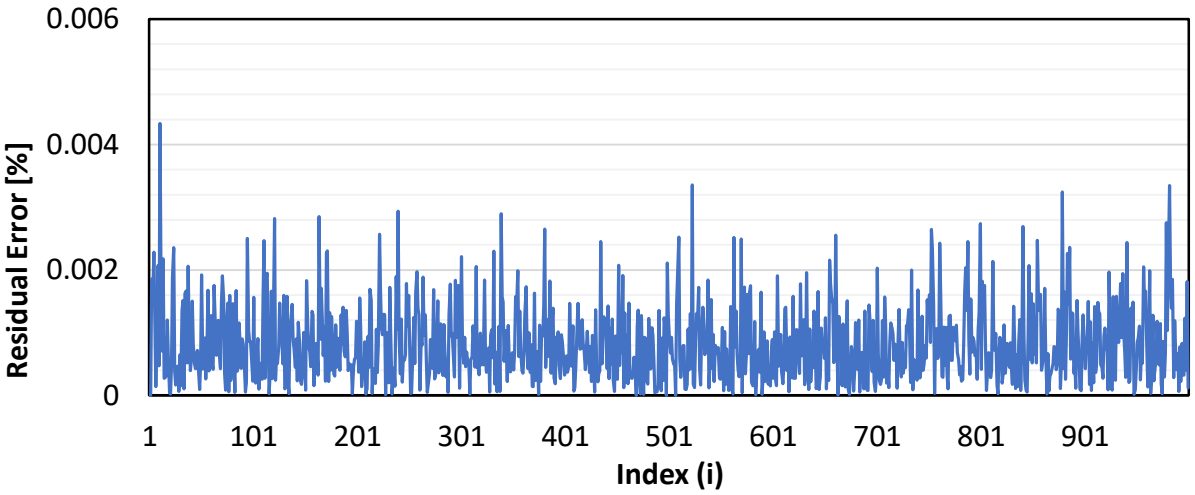


Figure 3.5: Residual Error of Centrifugal Pump OLS Test

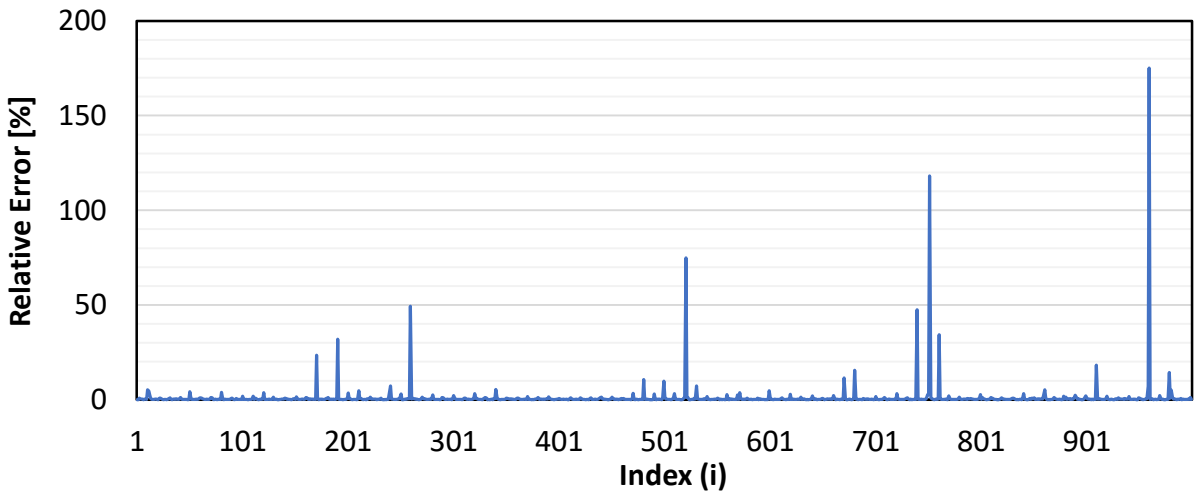


Figure 3.6: Relative Error of Centrifugal Pump OLS Test

As seen in the figures above, the relative error aids in pinpointing the index points with the highest deterioration in performance, regardless of scale as seen in the case of the residual error.

[9] Overfitting reflects the model's inability to discern between genuine underlying patterns and random variations present in the training data. an overfit model may perform well on the training data but fails to generalize effectively to new, unseen data

3.2.2 Principle of Identification Process

The main basis of the proposed solution, described in Figure 3.7, are Error Prioritization and Normalization, Memory/Time Cost Reduction, Median Regression and Relative Error Observation. The primary aim is to reduce the memory costs and runtime of identification algorithms and to maintain or improve algorithm efficiency and precision.

This reduction is to ensure proper functioning in a constraint simulation environment, mainly on the smartphone platform. For these reasons, the first focus of the algorithm is to introduce an alternative method that removes matrix inversion computation from the calculation of the precision matrix.

The observation of error variation is based on relative error calculation in order to minimize the influence of the scale variation on the integrity of the model. Hence, the observed error is calculated as follows

The algorithm would also prioritize differences in the error distribution in order to counter the small residual differences that result from OLS. This is done through the introduction of a function with a set of degrees of liberties. This prioritization is mean regression based with a median regression influence. In addition, a normalization factor is introduced in the algorithm in order to address heteroskedasticity. This is based on the work of Miller et al. [34] which highlights that if the transformed model produced by FGLS does not eliminate heteroskedasticity, standard errors and the resulting inference will be incorrect (if using standard errors estimated under the assumption of conditional homoskedasticity).

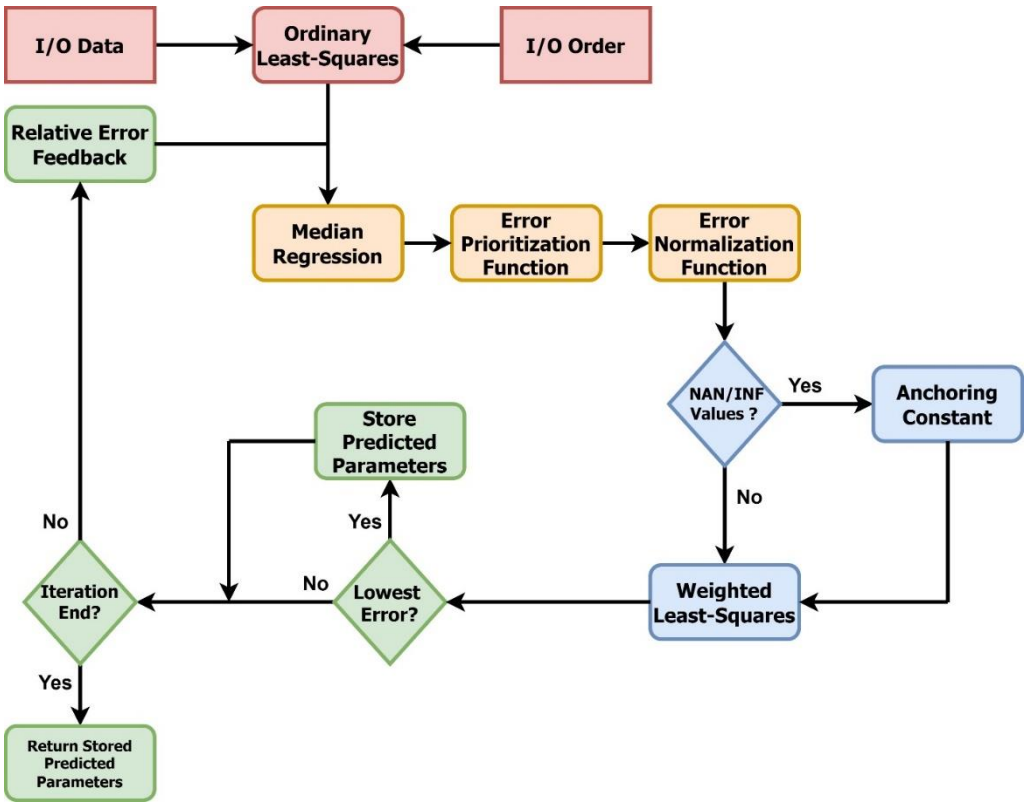


Figure 3.7: Proposed Algorithm Principle

3.2.3 Statement of Algorithm

To state the identification algorithm, the following functions are defined according to the value of the observed error given by (3.2). Let Ψ be a strictly decreasing function such that

$$\Psi(i)(\mathfrak{e}(i) - \bar{\mathfrak{e}}) < 0 \quad \text{for all } i \in [\![1, N]\!]$$

where $\bar{e}$ is the error average

Let also ζ be a strictly increasing function so that

$$\zeta(i) = 1 \Leftrightarrow \mathfrak{e}(i) = \bar{\bar{\mathfrak{e}}}$$

and

$$\zeta''(i) > 0$$

for all $i \in [\![1, N]\!]$ and where $\bar{\bar{\mathfrak{e}}}$ is the error median.

In what follows, ρ will be used to sort $\mathfrak{e}$ in increasing order such that

$$\mathfrak{e}\big(\rho(i)\big) \leq \mathfrak{e}\big(\rho(i+1)\big) \text{ for all } i \in [\![1, N]\!]$$

The temporary sorting of collected error data is to ensure that the defined functions vary based on error increase.

By using the definitions given above, the main proposed strategy for determining the matrix A (see(3.1)) is expressed by Algorithm 3.I.

Algorithm 3.I Improved Identification Algorithm

1. $\Omega \leftarrow I_N$ /* Initialization at identity matrix */
2. **Read** $\urcorner, Y, Y^P, X$ /* $\urcorner$: algorithm iterations */
3. **For** *k = 1* **to** $\urcorner$
4. **For** $i = 1$ **to** N
5. $\mathfrak{e}(i) \leftarrow \left|\frac{Y(i) - Y^p(i)}{\vartheta(Y(i), Y^P(i))}\right|$
6. $\Omega[\,\rho(i)\,,\rho(i)\,] \leftarrow \Gamma_\mu(\,\zeta(\,\rho(i)\,)\,\Psi\big(\,\mathfrak{e}(\rho(i))\big)\,)$
7. **End For** *i*
8. $A(k) \leftarrow Y\Omega^{-1}X^T(X\Omega^{-1}X^T)^{-1}$
9. **End For** *k*
10. **Return** $A(k)$ **for** $\bar{\mathfrak{e}}(k) = \bar{\mathfrak{e}}_{min}$ /* $\bar{\mathfrak{e}}_{min}$: *minimum value in* $\bar{\mathfrak{e}}$ */

The precision matrix is therefore defined as a diagonal matrix in a similar approach to the WLS method. The function Γ_μ is introduced to ensure that $\Omega[i, i\,] \neq 0$ and will be defined hereafter.

In the sequel, the aim is to analyze the impact of used functions, i.e. Ψ, ζ and Γ_μ, and their combinations in the proposed approach. More precisely, we consider three cases, noted ***C1,C2***

and $C3$ respectively. The primary difference between each combination is the parameter of the study that is emphasized and the analysis of its influence on the resulting error, such as the choice of the implementation of the maximum error or the standard deviation in the proposed combinations.

The objective of these three cases is to test several forms of calibration and the validity of the proposed solutions. For ζ , the difference between the cases is to test the impact of including median in the regression and the functions are defined as

$$\zeta^{C1}(i) = \frac{i}{N-i} \text{ for } i \in [\![1, N]\!]$$

and

$$\zeta^{C2}(i) = \zeta^{C3}(i) = \begin{cases} \dfrac{N\, i}{2\,(N-i)^2} & \text{for } i \in [\![1, \left\lfloor \frac{N}{2} \right\rfloor]\!] \\ \dfrac{i}{N-i} & \text{for } i \in [\![\left\lfloor \frac{N}{2} \right\rfloor, N]\!] \end{cases}$$

where $\lfloor . \rfloor$ denotes integer part and $\left\lfloor \frac{N}{2} \right\rfloor$ represents the median index

For Ψ , the objective is to test if the standard deviation of the error would be more efficient for normalization than the maximum error and is defined as

$$\Psi^{C1}(i) = \Psi^{C2}(i) = \frac{\bar{\epsilon} - \epsilon(i)}{\epsilon_{max}} \text{ for } i \in [\![1, N]\!]$$

and

$$\Psi^{C3}(i) = \frac{\bar{\epsilon} - \epsilon(i)}{\boldsymbol{\sigma}(\epsilon)} \text{ for } i \in [\![1, N]\!]$$

where $\boldsymbol{\sigma}(\epsilon)$ is the standard deviation of the error

As for Γ , it is defined as

$$\Gamma_{\mu}^{C1}(x) = \Gamma_{\mu}^{C2}(x) = \begin{cases} x + \mu & \text{for } x \neq -\mu \\ \mu & \text{for } x = -\mu \end{cases} \quad \text{where } \mu \in \mathbb{R}$$

and

$$\Gamma_{\mu}^{C3}(x) = \begin{cases} x & \text{for } |x| \geq \mu \\ \mu & \text{for } 0 \leq x < \mu \\ -\mu & \text{for } -\mu < x < 0 \end{cases} \quad \text{where } \mu \in \mathbb{R}$$

3.3 Comparison of Proposed Methods

In this section, a comparative study is proposed on four practical processes for highlighting the performances of the proposed approach by comparing it to some standard ones, summarized in Figure 3.9.

In order to assess the validity of the proposed algorithm, several tests are done on varied systems where the input order and output order are varied per test. The first segment is done with the chosen literature methods and the one with the lowest observed mean error is chosen. The second segment is done with the proposed algorithm in comparison to the chosen literature method.

CPU Usage is observed because of the targeted platform for our work which is the smartphone platform. This is to ensure that the proposed algorithm performance is optimized for the targeted constraint simulation environment. By using CPU usage, it is a common observation point as all of the tests are done on the same device. In addition, relative error is observed in order to ensure that the studied results are not dependent on the scale and size of the tests. This parameter is the primary indicator of the algorithm's output precision.

Max Relative Error is also observed primarily to see if the tested algorithms result in any frequent aberrant observation that influences the result. In addition, it also offers an observation of the impact of these observations on the mean relative error.

Figure 3.8 shows the chosen variables for comparison within the context of constraint simulation.

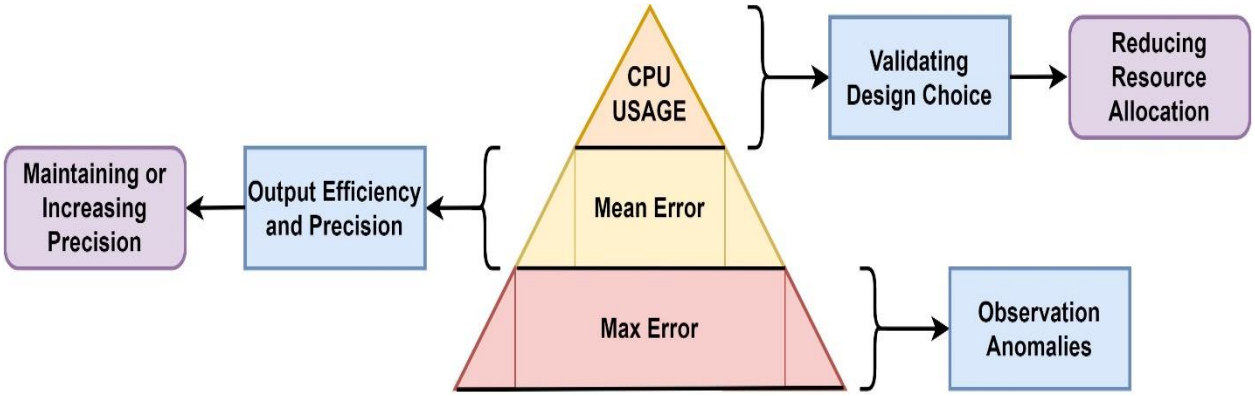


Figure 3.8: Observed Variables for Identification Process

CPU usage is the primary parameter for observation because of the focus on a constraint simulation environment. The main objective is to develop an algorithm that can be simulated without any risk of performance deterioration or operation freeze. However, algorithm

precision must also be maintained or improved. Therefore CPU usage observation is to verify the validity of the design choices and the mean error is to measure algorithm performance. Maximum Error is observed to check for observation anomalies.

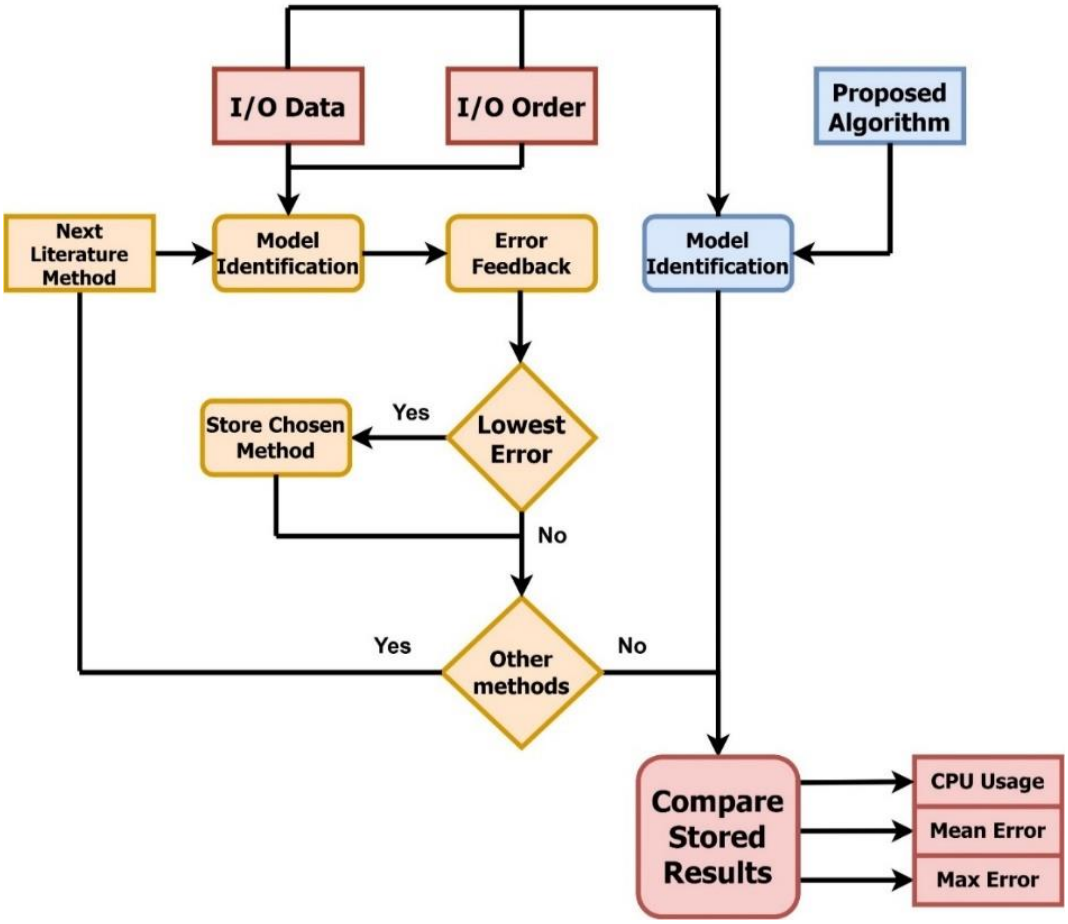


Figure 3.9: Comparison Process

3.3.1 Considered Process and Used Data

This section presents the setup of the data used for testing based on the simulation of set mechanical/electrical systems. The corresponding Input/Output data is described alongside the mathematical models that are used for the identification process.

3.3.1.1 Four process for Testing

The systems used here are the previously defined Centrifugal Pump, RLC circuit (see Figure 3.10), DC Servo Motor (see Figure 3.11) and Low-Pass Filter (see Figure 3.12).

The RLC circuit accepts a voltage signal as Input and a current signal is observed as the Output. It is composed of a resistor, inductor and capacitor. The relationship between the input voltage and the observed output current is linear. The RLC circuit is a standard concept that is widely spread in electrical studies and practices.

The third system is a DC servo motor. It takes a voltage signal as Input and the angle of rotation as Output. This system is typically used in various applications that require accurate and controlled linear motion such as in robotics.

The last system is a Low-Pass Filter which is a linear electrical system that represents a first order low-pass filter. It is considered linear in a specific range of input. The input to the system is a voltage signal and the output is a current signal.

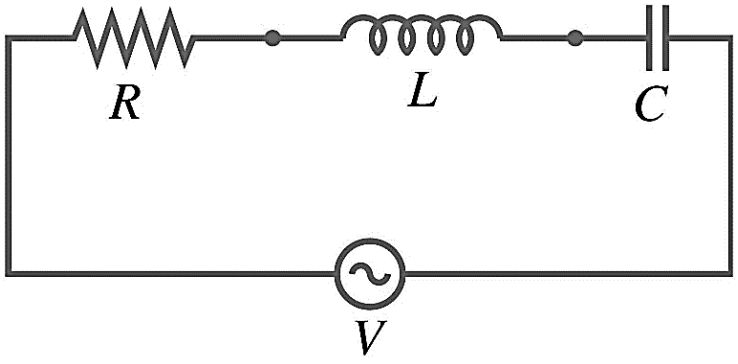


Figure 3.10: RLC Circuit Scheme

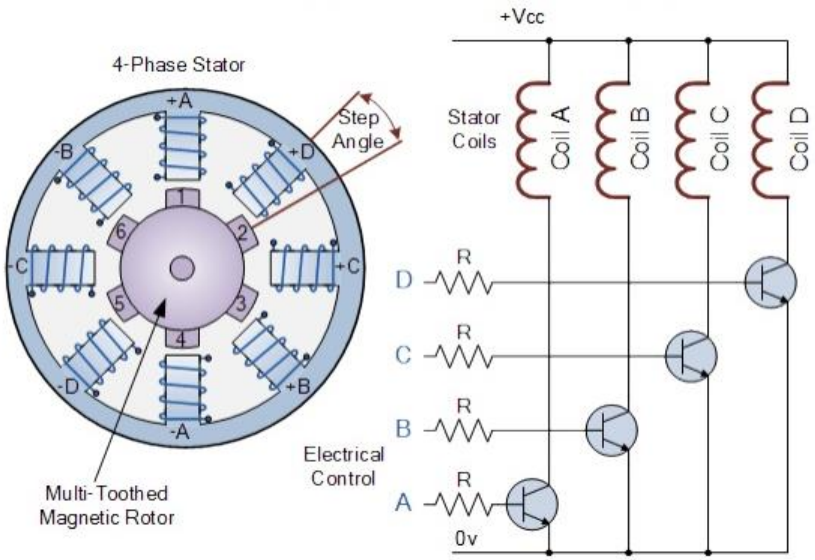


Figure 3.11: DC Servo Motor Design

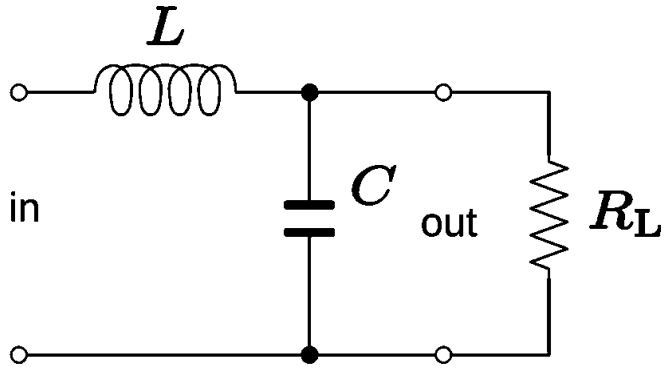


Figure 3.12: Low-pass filter Scheme

3.3.1.2 Used Input-Output Data for Testing

Each system is chosen to cover a specific range of the observed mean error and is replicated through Matlab for Simulation to be used for comparison. The simulation model of RLC Circuit and DC Servo Motor are given in [120], [121] respectively. The design and simulation of Low-pass filter is given in [122]. For each system, three tests are conducted where they differ by the input order and output order. The obtained Input-Output data for each system is shown below with RLC Circuit in Figure 3.13, DC Servo Motor in Figure 3.14, Low-pass Filter in Figure 3.15.

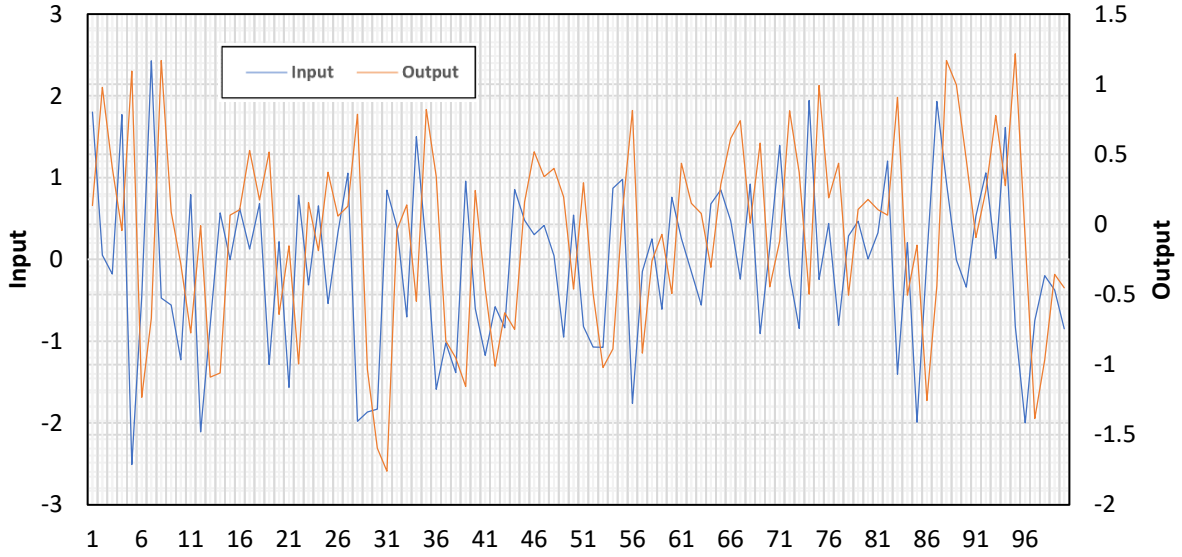


Figure 3.13: I/O Data of RLC Circuit

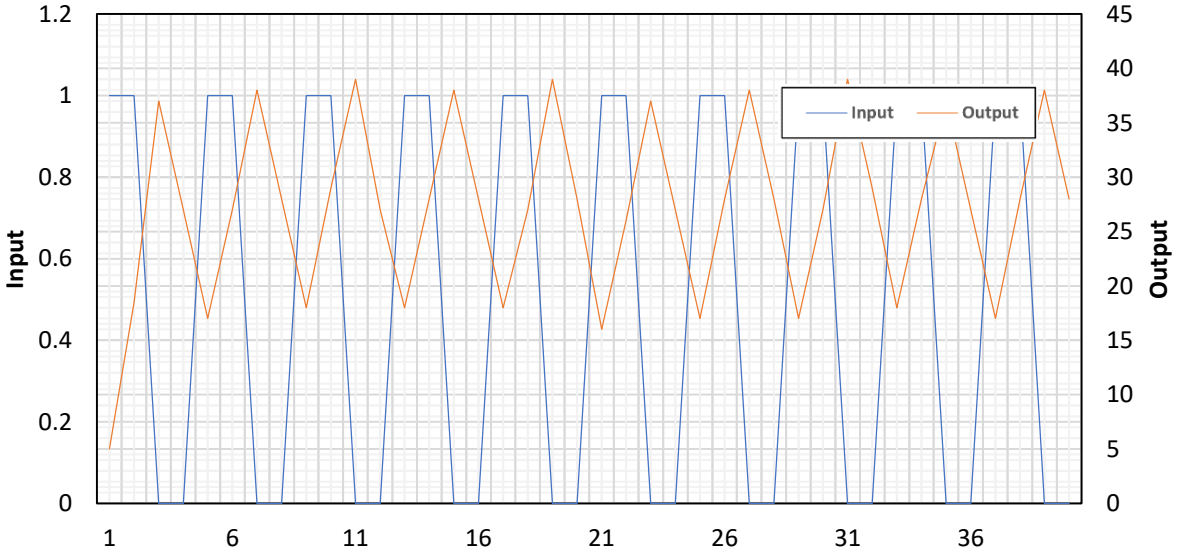


Figure 3.14: I/O Data of DC Servo Motor

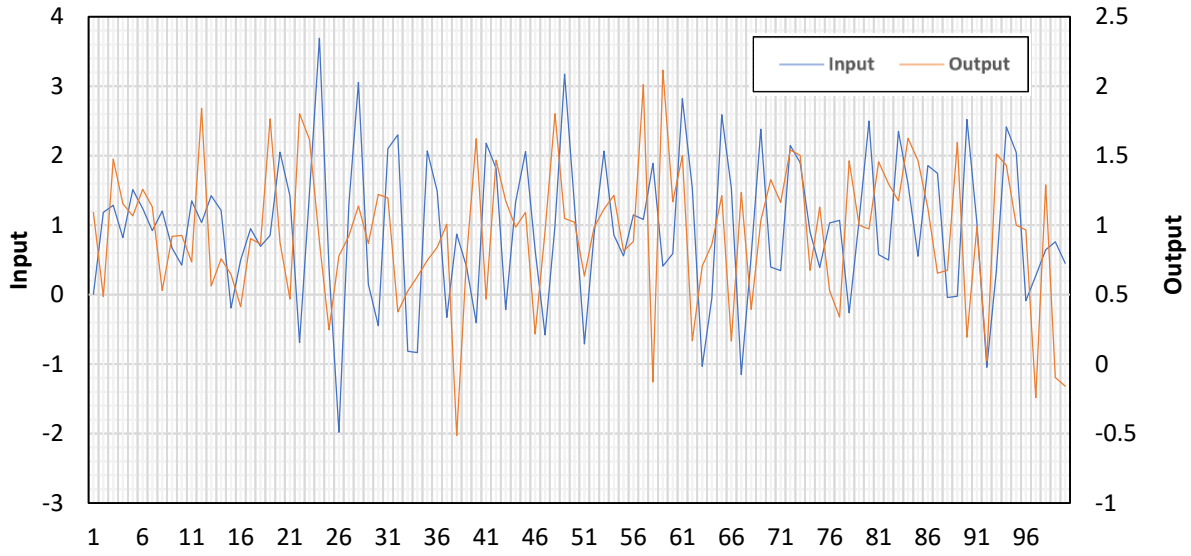


Figure 3.15: I/O Data of Lowpass Filter

3.3.1.3 Test Setup

The tests focus on the identification of an ARX model that represents a given system and is defined as

$$y(t) = a_1\, y(t-1) + \cdots + a_i\, y(t-i) + b_1\, u(t-1) + \cdots + b_j\, u(t-j) \tag{3.4}$$

where $a_1 \ldots a_i$ and $b_1 \ldots b_j$ are the ARX output and input parameters respectively and i, j the ARX model orders.

In order to satisfy (3.1), the model in (3.4) is translated into

$$\begin{bmatrix} y_{Max(i,j)} \ldots y_N \end{bmatrix} = \begin{bmatrix} a_1 \ldots a_i & b_1 \ldots b_j \end{bmatrix} \begin{bmatrix} y_{Max(i,j)-1} & \cdots & y_{N-1} \\ \vdots & \ddots & \vdots \\ y_{Max(i,j)-i} & & y_{N-i} \\ u_{Max(i,j)-1} & & u_{N-1} \\ \vdots & \cdots & \vdots \\ u_{Max(i,j)-j} & & u_{N-j} \end{bmatrix} \tag{3.5}$$

An ARX model is defined based on the set input and output orders. The first indicates the number of parameters that represent the input and the second indicates the number of parameters that represent the output.

3.3.2 Suitable Combination determination

The first round of tests concerns the comparison between the different versions of the proposed algorithm. Figure 3.16 and 3.17 shows the result of these comparisons.

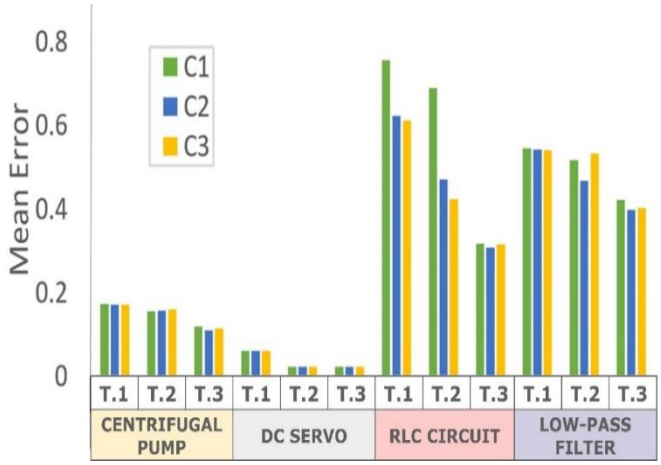


Figure 3.16: Mean Error Comparison between C1,C2,C3

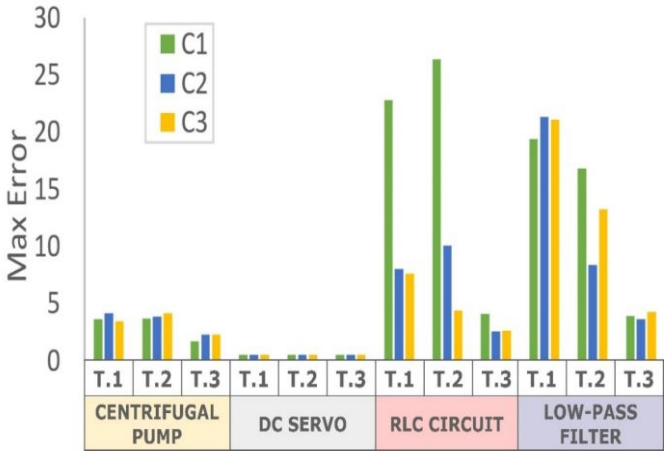


Figure 3.17: Max Error Comparison between C1, C2, C3

The observed data does not establish any clear combination that shows the best result in all of the tests. Because of this lack of a clear best, the standard deviation of the mean error is studied where

$$\boldsymbol{\sigma_{C1}} = 0.138,\ \boldsymbol{\sigma_{C2}} = 0.00946 \text{ and } \boldsymbol{\sigma_{C3}} = 0.122$$

Clearly, the errors observed by $C2$ are the most consistent. Based on max error, $C3$ has shown overall better results. However, $C3$ has shown significant variation in performance based on mean error observation. In addition, Max Error observations are largely unstable because it can easily be affected by outliers in linear systems and even more in the case of ARX models. Thus it is harder and riskier to adopt a method that is purely based on Max Error specifically because of the impact of outliers. For the reasons above and primarily the observed consistency in results based on mean error, $C2$ is recommended as the model of choice for further testing.

3.4 Comparison with Literature Methods

In this section, the proposed algorithm is compared with the literature method that shows the lowest mean error result for each test. The first chosen set of literature methods are the heteroscedasticity covariance estimators developed by J. MacKinnon, H. White [36] [37] noted hereafter HC0,HC1,HC2 and HC3 and Cribari-Neto, namely HC4, later on [38] which addresses non-constant variance in regression analysis. The second method is the CLM method [39] and is based on the study of autocorrelation in the error residuals. The last tested method is the HAC method [40] which adjusts covariance estimates to enhance the reliability of the approximation.

3.4.1 Performance of Chosen Methods

The literature methods used in these tests have been chosen because they offer a general approach. As highlighted by White and Mackinnon [37], each method has a set of advantages and disadvantages that vary based on given system and scenario.

However, one of the main common points between these methods is the observed CPU Usage. This is noticed in Figure 3.18 which shows the CPU usage and mean error for the chosen literature method with the lowest mean error. While memory and time costs are considered difficult to specifically calculate because of all of the variables in a given CPU, the observed CPU usage is with a relatively high-end desktop processor. Therefore this usage would only deteriorate performance even further when observed in the smartphone simulation environment. In addition, all of these methods have shown high CPU usage in some if not all of these tests and none of the methods showed a definitive lower usage. Thus a clear single candidate for the lowest CPU usage method cannot be specified. It is for these reasons that later on in the comparison, the literature method with the lowest observed mean error per test is chosen for comparison and no single method is chosen for all of the tests.

Table 3.1 shows the chosen literature method based on lowest mean error. This is observed by testing these methods on the same platform.

Centrifugal Pump	**T.1**	CLM
	T.2	HC0
	T.3	CLM
DC Servo Motor	**T.1**	HAC
	T.2	HC2
	T.3	HC3
RLC Circuit	**T.1**	HAC
	T.2	HC3
	T.3	HC0
Lowpass Filter	**T.1**	HC1
	T.2	HAC
	T.3	CLM

Table 3.1: Chosen Literature Method for each test

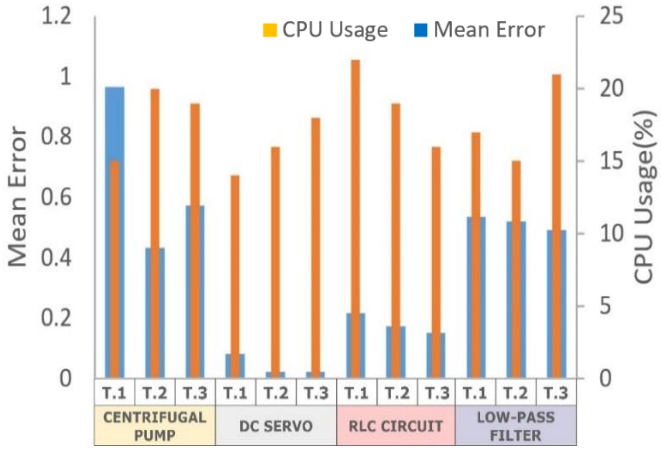


Figure 3.18: CPU Usage of Literature Methods

3.4.2 Comparison with Proposed Method

Figure 3.19 shows the comparison results between the proposed algorithm and the chosen literature methods.

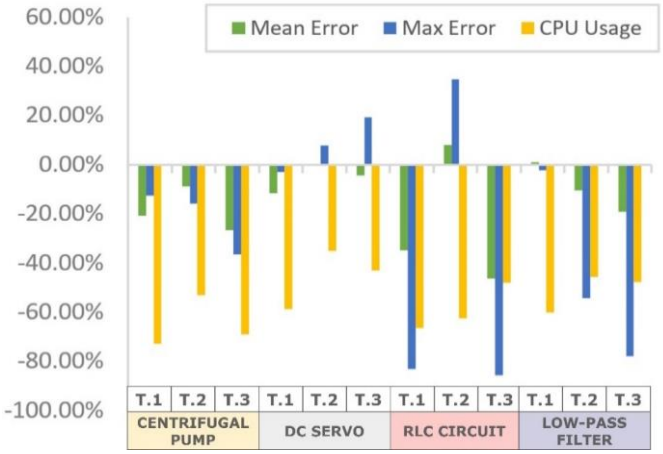


Figure 3.19: First Comparison with Literature Methods

Based on the results provided in Figure 3.19, the proposed algorithm succeeded in significantly reducing the CPU usage of the identification process. Concerning the other observed factors, the mean relative error is lower or equal in 10/12 tests whereas the max error is lower or equivalent in 9/12 tests.

The observed maximum error has shown an increase in variation in three tests. This is because of the potential instability of maximum error observation in the identification process and in ARX identification in specific. This instability is due to the risk of cascading error in ARX models.

Concerning mean error, RLC Circuit T.2 has shown an increase in mean Error after using c_2. However, based on other observations from several rounds of tests, Figure 3.20 shows the mean error results for Test 8 for all of the concerned methods.

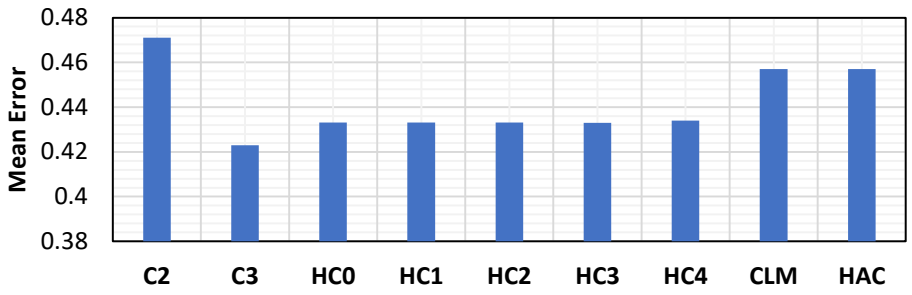


Figure 3.20: RLC Circuit T.2 Mean Error Results

The Mean Error observations for RLC Circuit T.2 show that ***C3*** offers better precision than ***C2*** and all of the literature methods. However, this is not the case in all of the other tests. This variation in results in Test 8 may imply the need for more degrees of liberty in the algorithm model in order to provide more room for calibration as the primary difference between ***C2*** and ***C3*** was in calibration and not in the main function ζ. The presented results show the potential in the basis of the algorithm but indicate that there is a possibility to improve further. For these reasons, the next attempt would be the introduction of a generalized form of the proposed algorithm with variable parameters that can adapt to a given system.

3.5 Generalized Form of Proposed Algorithm

This generalized form is based on ***C2*** and aims to improve the adaptability of the method. The function γ is changed with the addition of degrees of liberty a_v, b_v where $a_v, b_v \in \mathbb{R}$ such that

$$\zeta_v(i) = a_v \frac{i}{(N-i+1)^v} + b_v \tag{3.6}$$

where $v \in \mathbb{N}$ and $i \in [\![1, N]\!]$

Two specific variables for v are introduced, noted m and p. The separation between m and p is based on a chosen element of the presented data noted N_0 where $1 < N_0 < N$. Note that in ***C2***, N_0 is chosen as the median of the error array. This separation allows increasing the focus on the higher set of errors. In addition, the following restraints are placed where

$$\zeta_p(1) = \alpha, \zeta_p(N_0) = \zeta_m(N_0) = \beta, \zeta_m(N) = \delta \text{ and } \alpha \leq \beta \leq \delta$$

where $\alpha, \beta, \delta \in \mathbb{R}^+$ are preset and a_p, b_p, a_m, b_m are calculated based on these set restraints, obtaining

$$a_i = \begin{cases} \dfrac{\alpha - \beta}{N^{-p} - N_0(N+1-N_0)^{-p}} & \text{if } i = p \\ \dfrac{\beta - \alpha}{N_0(N+1-N_0)^{-m} - N} & \text{if } i = m \end{cases}$$

and

$$b_i = \begin{cases} \dfrac{\beta N^{-p} - \alpha N_0 (N+1-N_0)^{-p}}{N^{-p} - N_0 (N+1-N_0)^{-p}} & \text{if } \; i = p \\ \dfrac{\delta N_0 (N+1-N_0)^{-m} - \beta N}{N_0 (N+1-N_0)^{-m} - N} & \text{if } \; i = n \end{cases}$$

The objective of this generalized function is to add more degrees of liberty to the mathematical model and in turn test their impact on the observed outcome. In addition, these parameters offer more flexibility to the setup of Ω.

Note that *c2* is a specific form of this generalized function with the parameters as

$$p = 2, m = 1, N_0 = \frac{N}{2}, a_p = \frac{N}{2}, b_p = 0, a_m = 1 \text{ and } b_m = 0$$

The comparison tests are repeated and focus on studying the variation between using the generalized form of the proposed algorithm and the literature method with the lowest mean error result. The primary focus was on the tests that have shown a higher mean error after using *c2*. Figure 3.21 shows the variation result.

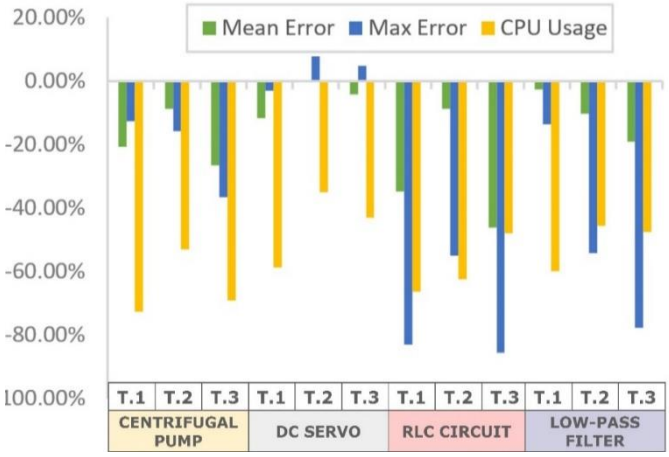


Figure 3.21: Variation after Using Generalized Algorithm results

The proposed generalized form maintains similar decreases in CPU usage while radically improving the Mean Error in all of the tests. A noticeable decrease in Maximum Error is noticed as well in most of the tests.

3.6 Conclusion

The proposed algorithm succeeded in lowering CPU usage while maintaining or increasing algorithm precision. The main drawbacks of this model is the need for degrees of liberty which can reduce the distance from the optimal solution. However, this addition adds a degree of customization for the process adapting the algorithm for each given scenario or environment. The next part of the work continues this mathematical optimization for larger processes in the domain of Control.

4 System Control Simulation Adapted for Constrained Environment

4.1 Introduction

This chapter concerns simulation in the field of Automatic Control, within the scope of Linear Quadratic Regulator (LQR) algorithms and their execution and implementation cost in a constraint simulation environment. It analyses the calculation costs of the control process and the effect of implementing order reduction methods for the matrices in study. The primary interest in LQR is because of its dependency on the Algebraic Riccati equation where solving this equation involves a large number of matrix computation both in potential number of iterations and number of operations per iteration, which can lead to high calculation costs primarily because of matrix inversion and multiplication [123]. The interest is using operations in a space with reduced dimension and hence to reduce computation time. This reduction is achieved through the implementation of order reduction methods. However, this will generate error comparing to the initial model. The challenge considered in this work is establishing a relationship between the computation time and the committed error for allowing to the user simulation management depending on the targeted goal. Figure 4.1 summarizes the workflow in this paper.

The work is organized as follows. The LQR approach is summarized in the second section. The concept of Matrix Order Reduction Methods is introduced in the third section. The simulation process and the results are shown in the fourth section. The fifth section explores extrapolation of the found results to other machines. In the sixth section, the relationship between the reduction of calculation time and precision loss is analyzed. The seventh section shows the use of the the proposed models and results in a case-study in the domain of Adaptive Optics. The final section offers a conclusion of the work.

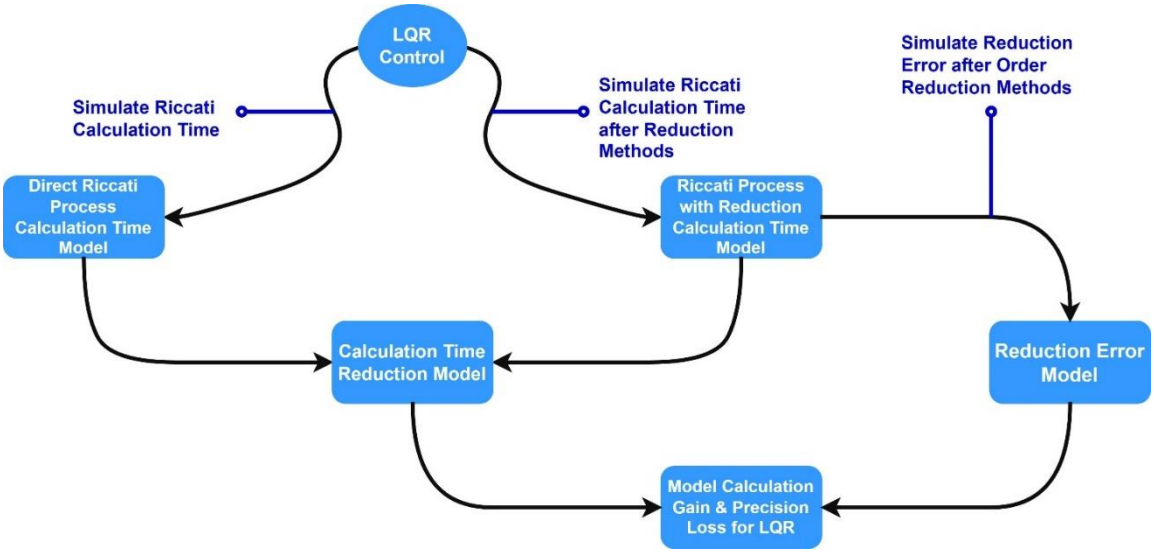


Figure 4.1: Planned workflow for the conducted tasks

4.2 Linear Quadratic Regulator Concept

The process of designing LQR is a well-established problem in control theory. It involves determining the state feedback gain that optimizes a quadratic cost function related to the plant's state and input variables [124]. This gain calculation necessitates solving a *Riccati* equation and is expressed as a function of the plant's state-space model. Consider the discrete system with input u, output y and state-space variable x such as

$$\begin{cases} x(i+1) = Ax(i) + Bu(i) \\ \quad y(i) = Cx(i) + Du(i) \end{cases} \tag{4.1}$$

where $i \in \mathbb{N}^*$, $A \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{p\times n}$ are the state space parameters and $B \in \mathbb{R}^{n\times m}$, $D \in \mathbb{R}^{p\times m}$ are input matrices with $n, m, p \in \mathbb{N}^*$

In the LQR approach, a main step for control synthesis is mainly based on solving the *Riccati* equation defined by

$$P = A^T PA - (A^T PB)(R + B^T PB)^{-1}(B^T PA) + Q \tag{4.2}$$

where $P \in \mathbb{R}^{n\times n}$ is the unknown symmetric definite matrix to be determined, $R \in \mathbb{R}^{m\times m}$, $Q \in \mathbb{R}^{n\times n}$ are LQR state space and input matrices respectively

The analytical solution of (1.2) is generally unknown. The standard algorithm for approximating *P* the solution for the *Riccati* equation, i.e. (1.2), (see for instance [125]) is given by Algorithm I. A core part of the algorithm is the Schur Canonical Form (SCF) which can be stated as follows. For $A \in \mathbb{R}^{n\times n}$ there exists $F \in \mathbb{R}^{n\times n}$ a quasi-upper triangular matrix with its diagonal elements being the eigenvalues of A. F is used to obtain P through the use of QR decomposition as seen in Algorithm 4.I. The algorithm is used for the purpose of this study because it is considered the direct method for approaching this problem in comparison to other iterative alternatives such as Disk Function Methods, Newton Iterations and other [126].

It should be noted that the approximation of *P* requires several sets of matrix multiplication and inverse per iteration. The work of Simoncini et al. [127] indicates that the resulting large computational costs of solving the *Riccati* equation where the CPU time can easily fluctuate depending on the state size, i.e. *n*, and result in an exponential increase in calculation time for higher values of the parameter *n*.

Algorithm 4.I Schur Method for Direct *Riccati* Process

1. **Set** Z $\leftarrow \begin{bmatrix} A + BR^{-1}B^{T}A^{-T}Q & -BR^{-1}B^{T}A^{-T} \\ -A^{-T}Q & A^{-T} \end{bmatrix}$
2. **Set** $S \leftarrow Z$, $V \leftarrow I$, $Check \leftarrow 0$ /*S_{Ci} indicates the ith $column$ in S*/
3. **While** $Check = 0$ /*SCF*/
4. **Set** $F_{C1} = \frac{S_{C1}}{\sqrt{S_{C1}^{T}S_{C1}}}$
5. **For** $i = 2$ to $2n$
6. $F_{Ci} \leftarrow \frac{S_{Ci} - \sum_{j=1}^{i-1}(F_{Cj}^{T}S_{Ci})F_{Cj}}{\sqrt{(S_{Ci} - \sum_{j=1}^{i-1}(F_{Cj}^{T}S_{Ci})F_{Cj})^{T}(S_{Ci} - \sum_{j=1}^{i-1}(F_{Cj}^{T}S_{Ci})F_{Cj})}}$ /*QR Decomposition for S*/
7. **End For**
8. **Set** $F \leftarrow [F_{C1}, F_{C2} \ldots F_{Cn}]$, $H \leftarrow F^{T}S$
9. $S \leftarrow HF$, $V \leftarrow VF$
10. $Check \leftarrow 1$ /*$Check$ is used to monitor convergence*/
11. **For** $j = 1$ to $2n - 1$
12. **If** $|s_{j+1,j}| > \omega$ **Then** $Check \leftarrow 0$ **End If** /*ω is the SCF convergence threshold*/
13. **End For**
14. **End While**
15. $P \leftarrow V_{21}V_{11}^{-1}$ where $V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}$

4.3 Matrix Order Reduction Methods

In the following, G is used to denote the matrix associated to system (1) defined by

$$G = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{4.3}$$

Let k be an integer $k < n$. Order reduction methods are used to obtain an equivalent reduced model of order k associated to G_k defined by

$$G_k = \begin{bmatrix} A_k & B_k \\ C_k & D_k \end{bmatrix} \tag{4.4}$$

where $A_k \in \mathbb{R}^{k\times k}, B_k \in \mathbb{R}^{k\times m}, C_k \in \mathbb{R}^{p\times k}$ and $D_k \in \mathbb{R}^{p\times m}$

In the framework of reduction approach several works have been proposed. In this section two standard order reduction methods are introduced and the algorithms description is presented.

The goal is analysing the performances of each methodology for choosing the more adapted one in the considered context. More precisely, the first is *B*alanced *M*odel *R*eduction (*BMR*) [128] and the second is *S*ingular *P*erturbation *A*pproximation (*SPA*) [129].

More precisely, the aim is to assess the impact of order reduction methods on the calculation time of the *Riccati* process. In this study, three procedures are analyzed. The first is the *Riccati* process without order reduction method. The second is with *B*alanced *M*odel *R*eduction. The third is with *S*ingular *P*erturbation *A*pproximation. Algorithm 4.II expresses the *Riccati* process combined with a reduction approach, namely *B*alanced *M*odel *R*eduction or *S*ingular *P*erturbation *A*pproximation.

Algorithm 4.II Solving with Reduction Methods

1. **Read** k, $\mathcal{M}$ /* $\mathcal{M}$ denotes the chosen method namely *BMR* or *SPA**/
2. **For** $i = 1$ **To** 2
3. **If (**i = 1) **Then** $E \leftarrow A$ **Else** $E \leftarrow A^T$ **End If**
4. **Solve** $E\ L_i {L_i}^T + L_i {L_i}^T E^T + (2-i)BB^T + (i-1)\ C^T C = 0$ with L_i the unknown
5. **End For**
6. **Set** $L_{SVD} \leftarrow {L_1}^T L_2$ /*SVD Decomposition of ${L_1}^T L_2$*/
7. **For** $i = 1$ **To** 2
8. **If** $(i = 1)$ **Then** $Z \leftarrow L_{SVD} L_{SVD}^T$ **Else** $Z \leftarrow L_{SVD}^T L_{SVD}$ **End If**
9. **For** $j = 1$ **To** n
10. **Solve** $\lambda_{i,j}$, $H_{Ci,j}$ for $\det(Z - \lambda_{i,j} I) = 0$, $Z H_{Ci,j} = \lambda_{i,j} H_{Ci,j}$ /*$\lambda_{i,j}$, $H_{Ci,j}$ eigenvalues and eigenvectors*/
11. **End For** j , **For** i
12. **Set** $S \leftarrow I$, $H \leftarrow [H_{C2,1}, H_{C2,2} \ldots H_{C2,n}]$
13. **For** $i = 1$ **To** n
14. $s_{i,i} \leftarrow \lambda_{1,i}$
15. **End For**
16. $V \leftarrow S^{-\frac{1}{2}} H^T {L_2}^T, \bar{A} \leftarrow VAV^{-1}, \bar{B} \leftarrow VB, \bar{C} \leftarrow CV^{-1}$
17. $G \leftarrow \begin{bmatrix} \bar{A} & \bar{B} \\ \bar{C} & D \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & B_1 \\ A_{21} & A_{22} & B_2 \\ C_1 & C_2 & D \end{bmatrix}$ /* $A_{11} \in \mathbb{R}^{k \times k}$ */
18. **If (**$\mathcal{M}$ = BMR) **Then** $M \leftarrow 0$ **Else** $M \leftarrow 1$ **End If**
19. $A \leftarrow A_{11} - \mathcal{M}\ (A_{12} A_{22}^{-1} A_{21}), B \leftarrow B_1 - \mathcal{M}\ (A_{12} A_{22}^{-1} B_2)$
20. $C \leftarrow C_1 - \mathcal{M}\ (C_2 A_{22}^{-1} A_{21}), D \leftarrow \mathcal{M}\ (D - C_2 A_{22}^{-1} B_2)$
21. **Execute** Algorithm 4.I

The precision of the Direct *Riccati* Process is in relation to SCF convergence threshold i.e. ω. With matrix reduction methods, it is also in relation to the reduced matrix size i.e. k. For this work, the main focus is on the relation between the Reduced Matrix Size and *Riccati* precision. Because of this, the SCF convergence threshold is fixed at a value based on the threshold chosen by the corresponding *Matlab* function. A conventionally used value is adopted here because the precision of the SCF tends to be measured by Gerschgorin disks of the ordered Schur Form which is separate from the *Riccati* Process [130] and not directly dependent on the convergence threshold.

4.4 Computation Time Modelling

This section focuses on the modelling time for the *Riccati* process. All of the tests are done on the same machine, denoted Machine A, where its specifications are described in Table 4.1. It should be noted that the considered machines are classical ones and not chosen for specific characteristics. The goal to realize an analysis independently of the chosen machines. A main objective is developing a methodology for extrapolating obtained results to other kind of machines. This is why two machines are chosen "randomly", namely Machine A, for modeling, and Machine B for extrapolating.

Machine Characteristics	Numerical values
Machine A	
Base Speed	3.30 GHz
Sockets	1
Core	4
Logical Processors	8
Machine B	
Base Speed	2.60 GHZ
Sockets	1
Core	2
Logical Processors	4

Table 4.1: Description of Specifications for machines used

For each instance, a sample of N_S^A simulations is conducted and the presented value represents the average of the recorded simulation elements. It should be noted that the larger this parameter is chosen the more accurate will be the result. However, N_S^A is chosen as a balance between minimizing variability in observed results and maintaining feasible simulation time for experimentation. The parameters m and p are assumed to be fixed because they are not changed after matrix order reduction. Afterwards, a corresponding model for CPU Time is identified for each algorithm. More precisely, the model is considered as a polynomial function depending on the state size, i.e. n.

The identification process is a least-squares regression [33] which is conducted on the first part of the interval and validated over the second part. The chosen degree for the identified polynomial is based on the lowest recorded relative error while having a positive derivative in the studied interval.

The relative error is defined as

$$e(i) = \frac{|T_{CPU}(i) - \hat{T}_{CPU}(i)|}{T_{CPU}(i)}, i \in \mathbb{N}^* \quad (4.5)$$

where $T_{CPU}(i)$ and $\hat{T}_{CPU}(i)$ are the real and predicted CPU Time at the instance of observation i respectively.

4.4.1 Modelling for Riccati's equation solving

Let two integers N_s^{min} *and* N_s^{max}, with $N_s^{min} < N_s^{max}$, be the numbers of the maximum and minimum values for the considered state size, respectively, i.e., $n \in [\![N_s^{min}, N_s^{max}]\!]$.
Technically, a first goal is to determine a polynomial model $\hat{T}_{CPU}$ depending on n, i.e. state size, defined over the interval $[\![N_s^{min}, N_s^{max}]\!]$. The calculation time of the *Riccati* calculation process is dependent on the number of operations. However, the time needed for the execution of operations can vary because of machine perturbation or other factors. Therefore a filter is applied to the observed data in order to ensure that

$$\Delta\tilde{T}_{CPU}(n_1) \leq \Delta\tilde{T}_{CPU}(n_2) \iff n_1 \leq n_2 \quad (4.6)$$

where $\tilde{T}_{CPU}(i)$ is the smoothed CPU Time at the instance of observation i and $\Delta\tilde{T}_{CPU}$ is the corresponding discrete derivative

Several filters have been considered for this process. The Savitzky-Golay Filter (SGF) is chosen here [131]. A SGF is favoured in this scenario in comparison to other standard filters mainly Moving Average Filter (MAF) because of its general better performance in reducing output noise in comparison to MAF which is used to preserve sharp step response which are favoured in other applications but to be avoided in this case and can also lead to significant distortions in the output [132].

Considering $\tilde{T}_{CPU}$ the filtered data, the SGF process is expressed by Algorithm 4.III.

Algorithm 4.III Savitzky-Golay Filtering Process

1. **For** $i = 1$ **To** n
2. **Set** d_{sgf}, w_{sgf} /*d_{sgf}: polynomial degree of SGF, w_{sgf}: window half-width at each $T_{CPU}(i)$*/
3. **Set** A_{sgf} **such that** $\begin{bmatrix} T_{CPU}(b_l) \\ \vdots \\ T_{CPU}(b_u) \end{bmatrix}^T = A_{sgf} \times \begin{bmatrix} (b_l)^0 & \dots & (b_u)^0 \\ \vdots & \ddots & \vdots \\ (b_l)^{d_{sgf}} & \dots & (b_u)^{d_{sgf}} \end{bmatrix}$
4. $\tilde{T}_{CPU}(i) \leftarrow A_{sgf} \times \begin{bmatrix} {T_{CPU}}^0(i) \\ \vdots \\ {T_{CPU}}^{d_{sgf}}(i) \end{bmatrix}$
5. **End For**

The deviation between the observed and filtered data is defined as

$$d(i) = \frac{T_{CPU}(i) - \tilde{T}_{CPU}(i)}{\tilde{T}_{CPU}(i)} \tag{4.7}$$

Figure 4.2 depicts the result of the Savitzky-Golay Filter and the corresponding deviation

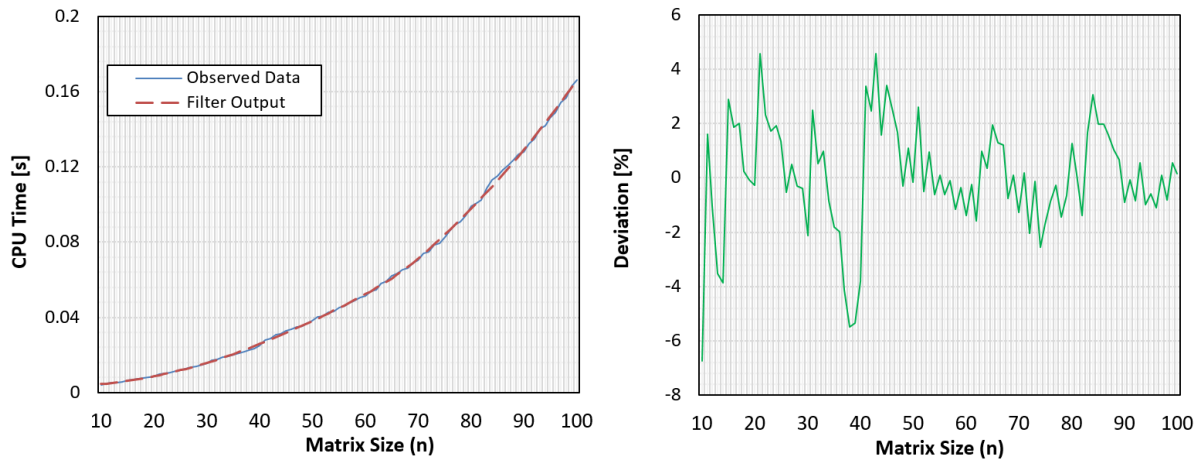


Figure 4.2: Savitzky-Golay Filter of *Riccati* Calculation Time

The next step of modelling the calculation time is the identification of the smoothed data. The calculation time of the *Riccati* calculation process is dependent on the number of operations executed. By using a result proposed in [125], an acceptable estimate is given by

$$O(T_{CPU}(n)) \approx \frac{188}{3} n^3 \tag{4.8}$$

However, this equation only offers an approximation of the floating point multiplications in the process and even though they are larger and more expensive than additions, these smaller operations are numerous and unrepresented. Because of this, a set of candidate models is chosen to balance between approximation and operations required and is defined as

$$\hat{T}_{\beta_1,\beta_2}(n) = \sum_{i=1}^{\beta_1} \left(a_i n^i + \beta_2 b_i n^{i-0.5}\right) + a_0 \,, \qquad n \in \mathbb{N}^* \tag{4.9}$$

where β_1, β_2 are parameters defining a model. When β_2 is equal to zero the model is a polynomial function. If equal to one, the model is a more general function that can be used for establishing an accurate model. For each subset, when β_1, β_2 are chosen, a_i and b_j will be parameters to be estimated. Furthermore, in this work, it is assumed, for obvious reasons, that the identified model is monotonous and increasing over $n \in \mathbb{N}^*$. For identification, the approach chosen is based on the least-square principle.

Therefore the process follows a constrained least-squares identification with the objective defined as

$$\begin{aligned} &\text{Minimize } \left\|\hat{T}_{\beta_1,\beta_2} - \tilde{T}_{CPU}\right\|^2 \\ &\text{such that } \hat{T}_{\beta_1,\beta_2}(n_1) \le \hat{T}_{\beta_1,\beta_2}(n_2) \text{ for all } n_1, n_2 \in \mathbb{N}^* \text{ with } n_1 \le n_2 \end{aligned} \tag{4.10}$$

A set of candidate models based on varying β_1, β_2 is tested. More precisely, since (1.8), β_1 is assumed to be less or equal to three. Finally, in this part, four structures are analyzed and compared for obtaining accurate model describing the behavior of CPU time depending on the state size. In identification step the interest is in ensuring a lower error between the model and the observed data. However, it is also important to beware of the well-known overfitting phenomena that can occur when the model is associated closely to the observation data. Hence, in this study, and in order to check for any potential overfitting of the model, the process is split into an identification phase and a validation phase where the relative error is measured for each. The first part of the process, i.e. identification phase, is conducted on a subset of the observed data, namely $[\![N_s^{min}, 0.1 p_s N_s^{max}]\!]$ where $p_s \in L_s^P$ such that $L_s^P = [\![2,9]\!]$ in this case, and the second part is on the other, namely $[\![0.1 p_s N_s^{max}, N_s^{max}]\!]$. Note that L_s^P was chosen as such for this work in order to cover the largest range of the data being used for testing but can be varied. The variation of p_s is to minimize the influence of the data range on the efficiency of the process in turn reducing the risk of overfitting. Figure 4.3 highlights a sample of this for several varying validation phase starts.

Table 4.2 presents the results of the different tests realized by varying the value for p_s. The results are presented by depicting relative errors. The last two lines - the average and standard deviation values - are given. The goal is estimating the quality of each considered structure. One can remark that the structure $\hat{T}_{3,0}$ realizes the better performances both in terms of identification and validation. Indeed, with this structure the average and standard deviation errors are lower. Besides, it should be noted that the structures $\hat{T}_{2,0}$ and $\hat{T}_{3,1}$ offer lower identification average error than $\hat{T}_{2,1}$ respectively. However, $\hat{T}_{2,1}$ delivers better performances in term of validation than the two other models. Finally, a protocol taking into account several analysis criteria must be determined for comparing and concluding in relevant way. Hence, in the next sections, an in-depth analysis is proposed for establishing an approach for modelling. This will be based on data recursive filtering and robustness analysis.

	$\hat{T}_{2,0}$		$\hat{T}_{2,1}$		$\hat{T}_{3,0}$		$\hat{T}_{3,1}$	
p_s	$e_i(\%)$	$e_v(\%)$	$e_i(\%)$	$e_v(\%)$	$e_i(\%)$	$e_v(\%)$	$e_i(\%)$	$e_v(\%)$
2	~ 0	3.34	~ 0	3.34	~ 0	3.34	~ 0	3.47
3	~ 0	3.50	~ 0	3.50	~ 0	3.50	~ 0	3.14
4	0.09	4.06	0.04	3.90	0.04	3.75	0.05	5.97
5	0.60	5.30	0.30	8.16	1.07	2.98	0.12	14.51
6	1.20	8.13	0.38	9.40	1.20	8.20	0.13	3.22
7	1.02	10.20	1.060	8.00	1.06	10.13	0.12	14.55
8	3.06	7.30	2.30	0.69	2.92	1.85	0.47	8.90
9	6.00	4.40	2.09	1.35	3.13	0.52	1.16	2.40
Average	1.50	5.77	7.71	4.79	1.18	4.28	2.56	7.02
Standard Deviation	2.08	2.49	0.94	3.29	1.25	3.23	0.39	5.08

Table 4.2: Identification process after single-time filtering

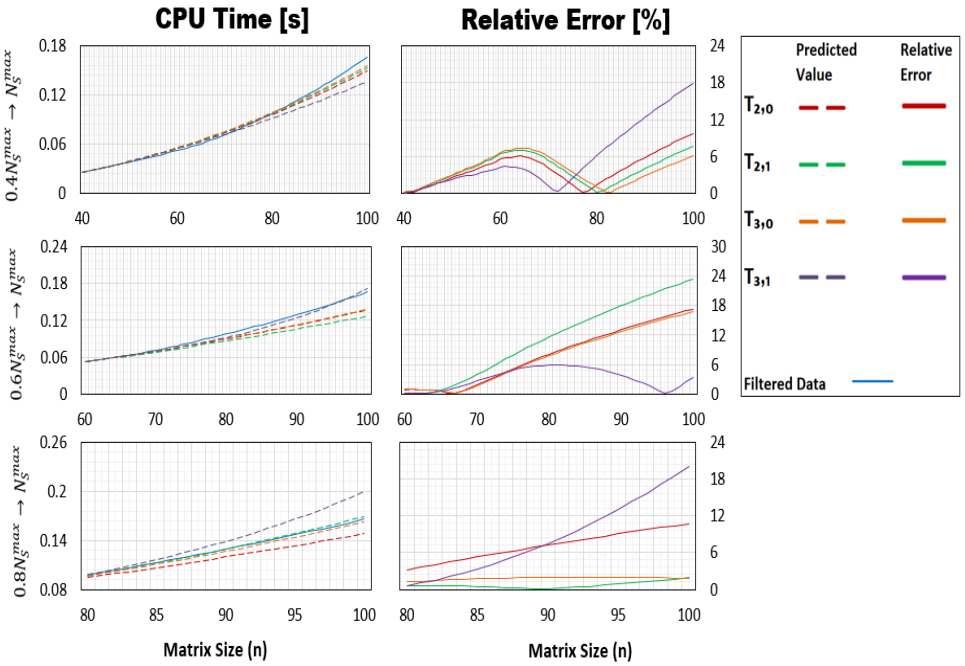


Figure 4.3: Predicted values and relative error

4.4.2 Recursive Filtering

Several processes have shown sub-optimal results in Table 4.2. Particularly, an increase both in identification and validation error is witnessed in several cases resulting in a random pattern of error variation in the tests. This section introduces the filtering process that is employed with the aim to counter these perturbations in the results.

4.4.2.1 Concept Introduction

In order to spot the reason behind this, an approximation of the derivative of the filtered data is studied where this approximation is defined as

$$\Delta\tilde{T}_{CPU}(i) = \tilde{T}_{CPU}(i) - \tilde{T}_{CPU}(i-1) \text{ for } i \in [\![N_s^{min}, N_s^{max}]\!] \tag{4.11}$$

The results of the derivative approximation after filtering are offered in Figure 4.4.

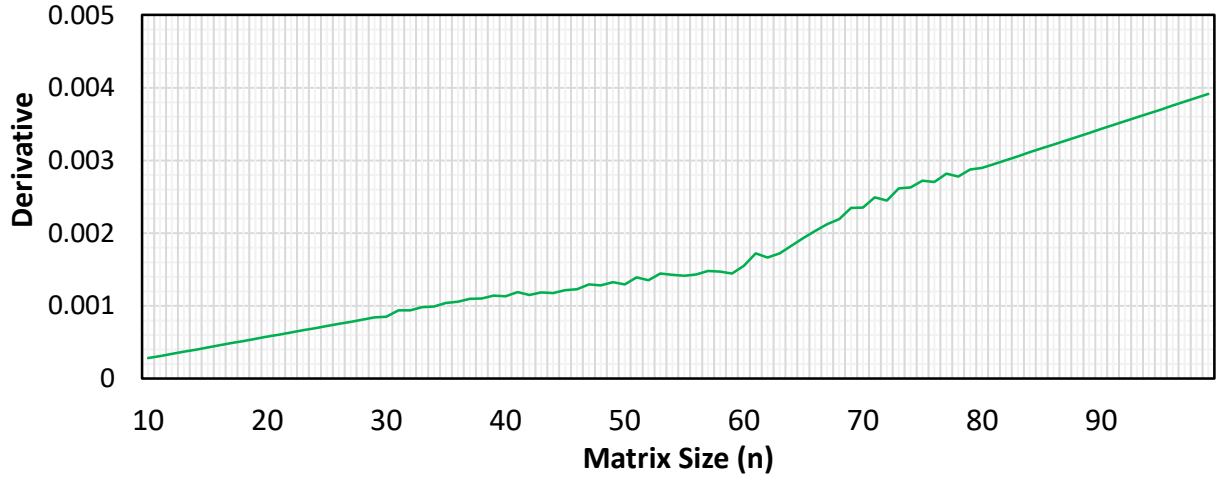


Figure 4.4: Derivative Approximation of Filtered Data

It should be noted that with a first filtering, the approximation shows a disturbance located in the middle of the curve. To overcome this disturbance, the goal is to realize a recursive filtering to smooth more the used curve.

For this, a SGF recursive filtering process is introduced inspired by similar approaches for other filters as in [133]. This process focuses on applying the same filtering process on a given observed data for several iterations where the input of the current iteration is the output of the previous one. Considering g_1 as the function representing the previously used filtering process where

$$\tilde{T}_{CPU} = g_1(T_{CPU}) \tag{4.12}$$

For $s \in [\![1, s_f]\!]$ where s_f is the maximum number of iterations, the recursive filtering is defined by

$$\tilde{T}^s_{CPU} = g_s(T_{CPU}) = g_1\big(g_{s-1}(T_{CPU})\big) \tag{4.13}$$

Figure 4.5 depicts the results of the derivative approximation after successive filtering processes. It should be noted that more the filtering is applied more disturbance affecting the derivative is vanished and hence curves are smooth.

Figure 4.6 shows the resulting filtered data. The direct difference in the derivative between a single filtering iteration and successive iterations is the loss of the perturbation observed during the first phase at the middle of the curve. This smoothness of the derivative should translate to less aberrations in the identification process.

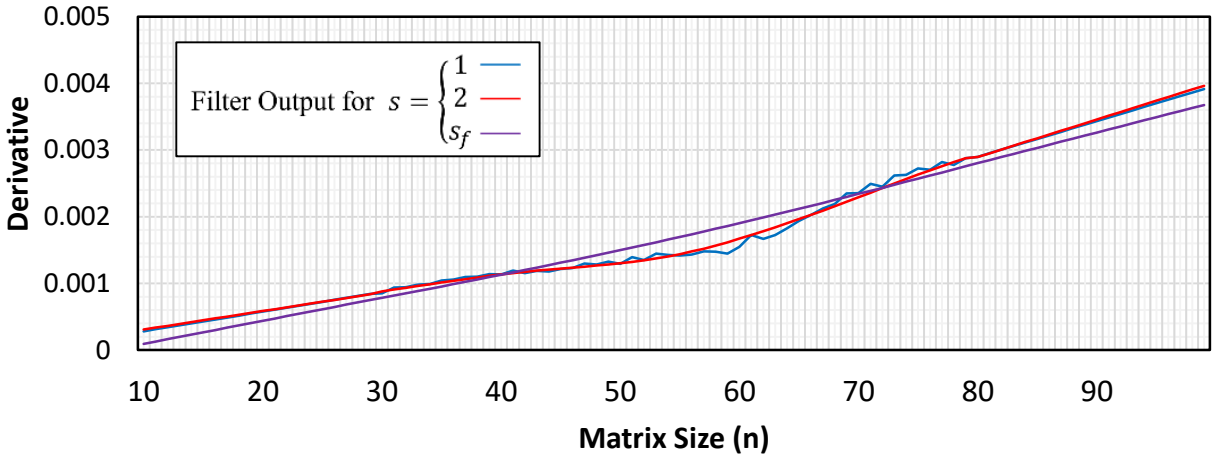


Figure 4.5: Derivative Approximation Based on Recursive Filtering Output

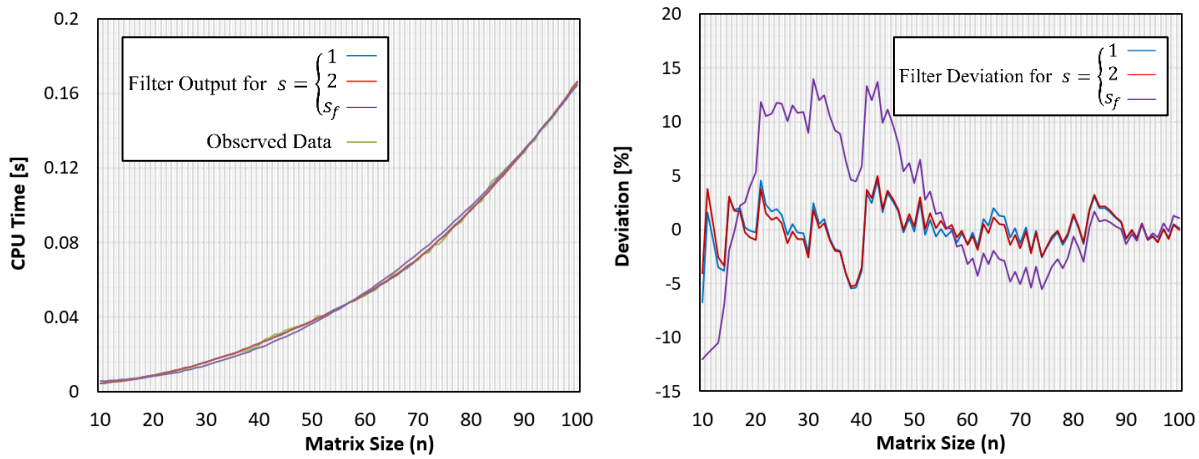


Figure 4.6: Deviation Based on Recursive Filtering

Finally, the next section is dedicated to modeling by using sequential filtering. The goal is to realize the modeling based on data that is free as maximum as possible of disturbances.

4.4.2.2 Modeling with Recursive Filtering

Based on the recursive filtering obtained data for maximum number of iterations, the identification process previously from Table 4.2 is repeated and the results are presented in Table 4.3. One can remark that all candidate models have experienced a decrease in recorded errors, both identification and validation, alongside lower aberrations in observations. While $\hat{T}_{2,1}$ and $\hat{T}_{3,1}$ deliver the lowest identification error in the tests, $\hat{T}_{2,1}$ offers the best validation error across all of the candidate models. However, $\hat{T}_{2,0}$ has the lowest standard deviation with regards to the validation error. This is why a robustness study is employed later on in order to verify the stability and sensitivity of the candidate models in face of perturbation or variation in the observation data.

	$\hat{T}_{2,0}$		$\hat{T}_{2,1}$		$\hat{T}_{3,0}$		$\hat{T}_{3,1}$	
p_s	$e_i(\%)$	$e_v(\%)$	$e_i(\%)$	$e_v(\%)$	$e_i(\%)$	$e_v(\%)$	$e_i(\%)$	$e_v(\%)$
2	~ 0	2.16	~ 0	2.16	~ 0	2.16	~ 0	2.40
3	~ 0	2.40	~ 0	2.40	~ 0	2.50	~ 0	2.40
4	0.02	3.09	0.01	3.40	0.01	3.53	0.01	1.75
5	0.06	3.40	0.03	2.16	0.52	2.64	0.01	1.82
6	0.30	3.20	0.08	1.13	0.14	1.54	0.01	0.85
7	0.73	2.70	0.14	0.42	0.23	0.66	0.43	7.96
8	1.29	2.02	0.17	0.05	0.30	0.13	0.01	0.53
9	1.93	1.40	0.16	0.10	0.30	0.10	0.10	2.24
Average	0.54	2.54	0.07	1.47	0.19	1.65	0.07	2.49
Standard Deviation	0.72	0.68	0.07	1.23	0.18	1.26	0.14	2.31

Table 4.3: Identification process after maximum iteration filtering

While Table 4.3 highlights an evident increase in the precision of the identification process, the main issue that arises is the increase in deviation as seen in Figure 4.6. This variation based on sequential filtering is analyzed where the absolute deviation, i.e. $|d(i)|$, is considered.

Figure 4.7 depicts the mean absolute deviation with respect to the number of filtering iterations conducted on the data. This shows that the chosen polynomial model later on can be modeled after filtering iterations lower than the maximum, particularly at the twentieth iteration.

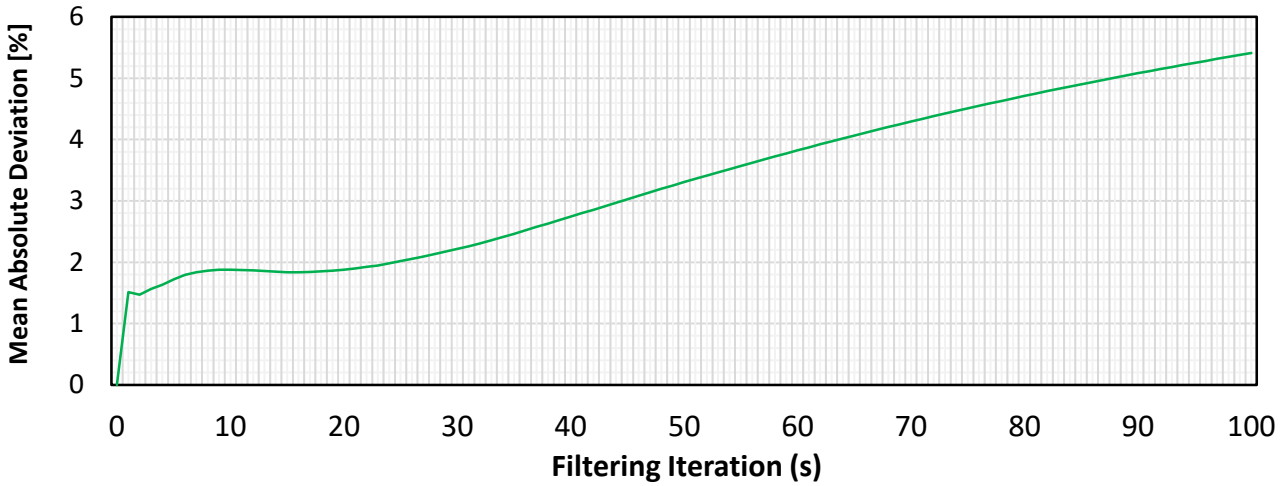


Figure 4.7: Variation of Mean Absolute Deviation

The resulting data from recursive filtering satisfy show a maximum deviation of five per cent from the original observation data, therefore satisfying the ninety five confidence interval for data extraction.

In order to find the proper balance between the deviation and the resulting error, let $e_{i,j}$ and $e_{v,j}$ respectively indicate the recorded identification error given a validation phase starting at $0.1\ p_s N_s^{max}$. Then, for a candidate model $\hat{T}_{i,j}$, the total errors, for identification and validation, expressed respectively by

$$\bar{e}_i = \frac{1}{|L_s^P|} \sum_{p_s \in L_s^P} e_{i,p_s} \tag{4.14}$$

and

$$\bar{e}_v = \frac{1}{|L_s^P|} \sum_{p_s \in L_s^P} e_{v,p_s} \tag{4.15}$$

where $|\ .\ |$ denotes the cardinality of a given set i.e. the number of elements it contains

Figure 4.8 presents the resulting errors for each candidate model per filtering iteration.

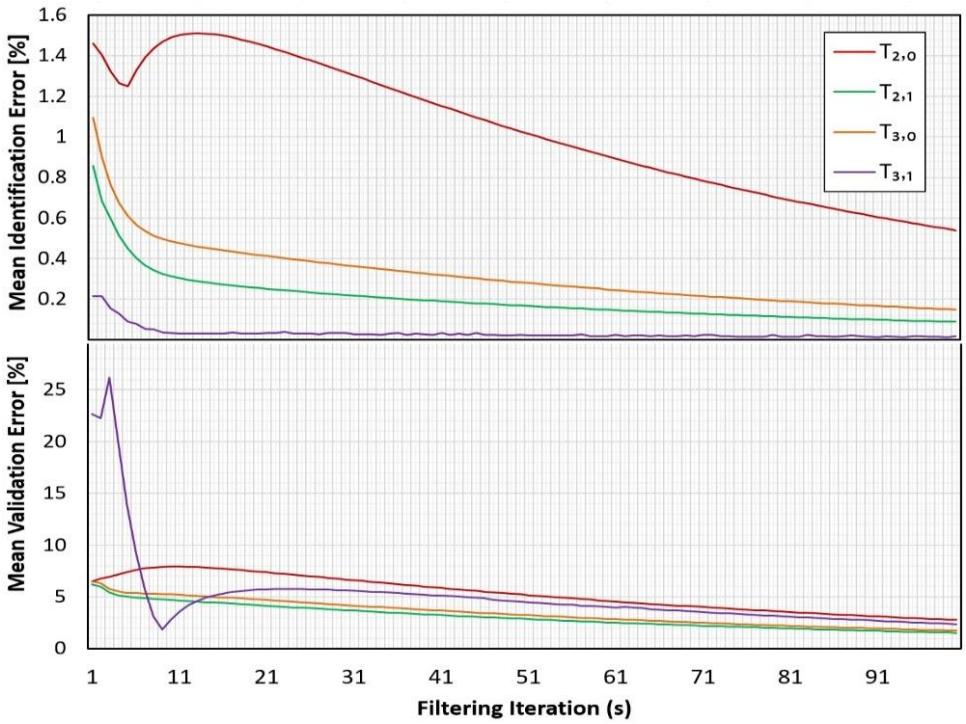


Figure 4.8: Mean Errors of Candidate Models

As expected, the recorded errors show a constant decrease per filtering iteration eventually leading to the results presented in Table 4.3 except at earlier filtering stages such as the case for $\hat{T}_{2,0}$ and $\hat{T}_{3,1}$. This is due to the presence of perturbations in the derivative of the observation data similarly to those in Table 4.2 and are only cleared after several iterations. This is also reinforced by Figure 4.5 with the observation on derivative approximations.

4.4.3 Robustness study

In the case of estimation, the identified model requires that the estimators change only slightly if the distribution of the observations is slightly altered from that used for the identification process [134].

Let $\hat{T}_{i,j}^{S}$ the identified candidate model corresponding to the observation data $\tilde{T}_{CPU}^{S}$.

Robustness would be evaluated based on variations in the observation data used for identification. In turn, the change in the estimated calculation time is monitored for $n \in [\![N_s^{min}, N_s^{max}]\!]$. This approach is based on the concept of Sensitivity Analysis where mathematical methods assess the sensitivity and robustness of a model output to the range of variation of an input. These methods typically involve calculating the output for a set of values that represent the possible range of the input [135]. This range is represented through two cases where the first is based on the correlation between the data and the second is based on stochastic variation.

In the sequel, c_R will be used for defining $T_r^s$and will take two values, namely zero or one, according to the case considered

$$T_r^s(i) = (1 - c_R)\tilde{T}_{CPU}^s(i) + c_R(T_r^{s_f}(i) + \omega_s) \tag{4.16}$$

where T_r^s is the observation used for identification and ω_s the corresponding perturbation at iteration $s \in [1, s_f]$

These two cases are chosen because they offer a feasible observation set whether it is due to variation in filtering iteration or perturbation in hosting processor performance.

For the case c_R is equal to zero, robustness is based on the variation of the deviation because this indicates a variation in observation distribution where

$$\left(\bar{d}_{s_1} \neq \bar{d}_{s_2} \Leftrightarrow \sum_{i=N_s^{min}}^{N_s^{max}} \frac{T_{CPU}(i) - \tilde{T}_{CPU}^{s_1}(i)}{\tilde{T}_{CPU}^{s_1}(i)} \neq \sum_{i=N_s^{min}}^{N_s^{max}} \frac{T_{CPU}(i) - \tilde{T}_{CPU}^{s_2}(i)}{\tilde{T}_{CPU}^{s_2}(i)} \right) \Rightarrow \exists j \text{ such that } \tilde{T}_{CPU}^{s_1}(j) \neq \tilde{T}_{CPU}^{s_2}(j)$$

For the case c_R equal to one, robustness is studied based on stochastic variation of observation data represented by ω_s.

To measure the change of the estimators, the relative deviation from the mean of recorded outputs is studied which, for a set matrix size, is defined as

$$\delta_n^s(\hat{T}_{i,j}) = \left| \frac{s_f \hat{T}_{i,j}^s(n) - \sum_{s=1}^{s_f} \hat{T}_{i,j}^s(n)}{\sum_{s=1}^{s_f} \hat{T}_{i,j}^s(n)} \right| \tag{4.17}$$

Therefore the developed algorithm would measure the deviation per each variation of the data used for the identification process as seen in Algorithm 4.IV.

Figure 4.9 offers a sample of the observation data with perturbation used for c_R equal to one and Figure 4.10 and 4.11 describe δ_s for each candidate model per iteration for the different values of c_R with the corresponding normalized mean of the relative deviation i.e. $\bar{\delta}$ in Table 4.4. The performance metrics are normalized in order to improve the probability of obtaining the global optimal solution when there are several local optimal solutions [136].

$\hat{T}_{3,0}$ and $\hat{T}_{3,1}$ result in a much larger deviation in comparison to the other models. This indicates the increase in stability at the higher degrees of the studied polynomials which is primarily due to the implementation of the constraints set in (1.10). $\hat{T}_{2,0}$ and $\hat{T}_{2,1}$ show the best result in each separate test with $\hat{T}_{2,0}$ offering better performance by a small margin. However, $\hat{T}_{2,0}$ resulted in higher errors in the previous tests. A cost function to be minimized is proposed afterwards in order to decide which model is optimal for calculation time model based on the combination of the tests conducted.

Algorithm 4.IV Robustness Testing of Candidate Models

1. **For** $c_R = 0$ **To** 1
2. **For** $s = 1$ **To** s_f /*Recursive Filtering Iterations*/
3. **For** $i = 1$ **To** n /*Filtering Process of Observation Data*/
4. $\tilde{T}_{CPU}(i) \leftarrow (1 - c_R) g_s\left(T_{CPU}(i)\right) + c_R \left(g_{s_f}(T_{CPU}(i)) + \omega_s\right)$
5. **End For** i
6. **For** $\beta_1 = 2$ **To** 3, **For** $\beta_2 = 0$ **To** 1
7. **Minimize** $\left\|\hat{T}_{\beta_1,\beta_2} - \tilde{T}_{CPU}\right\|^2$ /*Identification Process based Data Variation*/
8. **End For** β_1, β_2, s
9. **For** $s = 1$ **To** s_f, **For** $\beta_1 = 2$ **To** 3, **For** $\beta_2 = 0$ **To** 1
10. **For** $n = N_s^{min}$ **To** N_s^{max}
11. $\delta_n^s\left(\hat{T}_{\beta_1,\beta_2}\right) \leftarrow \left|\frac{s_f \hat{T}^s_{\beta_1,\beta_2}(N_C) - \sum_{s=1}^{s_f} \hat{T}^s_{\beta_1,\beta_2}(N_C)}{\sum_{s=1}^{s_f} \hat{T}^s_{\beta_1,\beta_2}(N_C)}\right|$
12. **End For** n
13. $\delta^S\left(\hat{T}_{\beta_1,\beta_2}\right) \leftarrow \frac{\sum_{n=N_s^{min}}^{N_s^{max}} \delta_n^s}{N_s^{max} - N_s^{min}}$
14. **End For** β_1, β_2, s
15. **End For** c_R

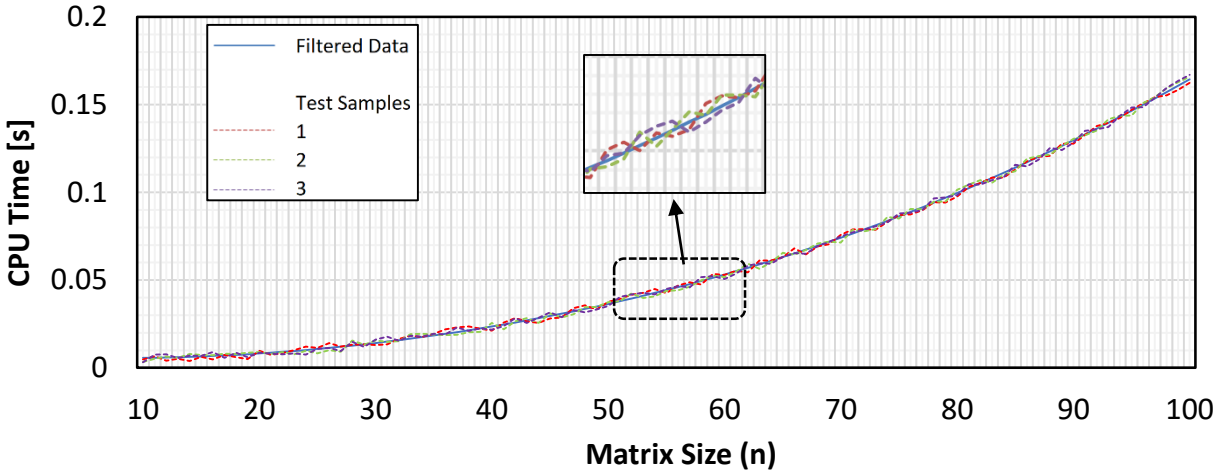


Figure 4.9: Sample of Data Sets used for Robustness Testing

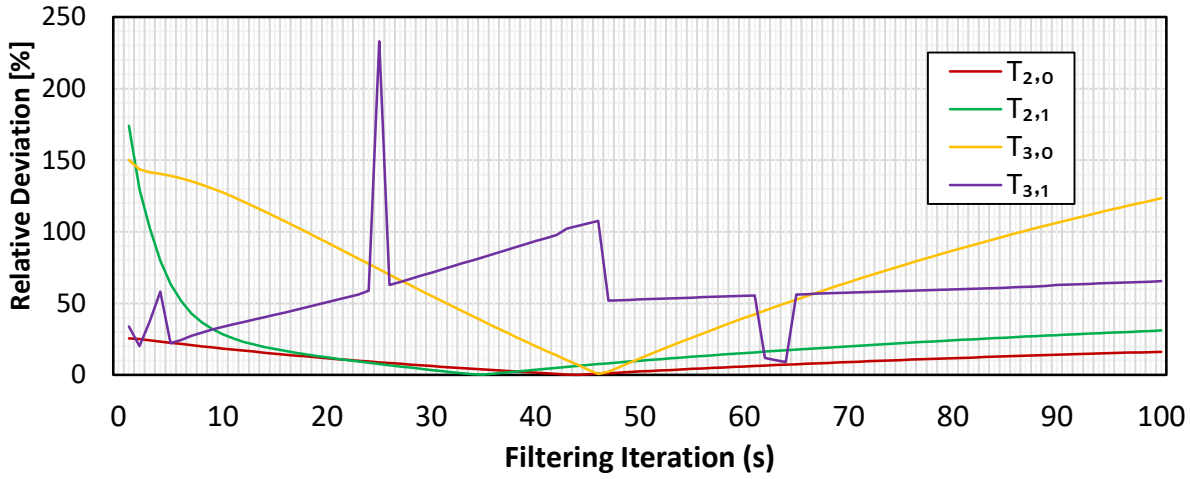


Figure 4.10: Relative Deviation Variation – First Robustness Testing

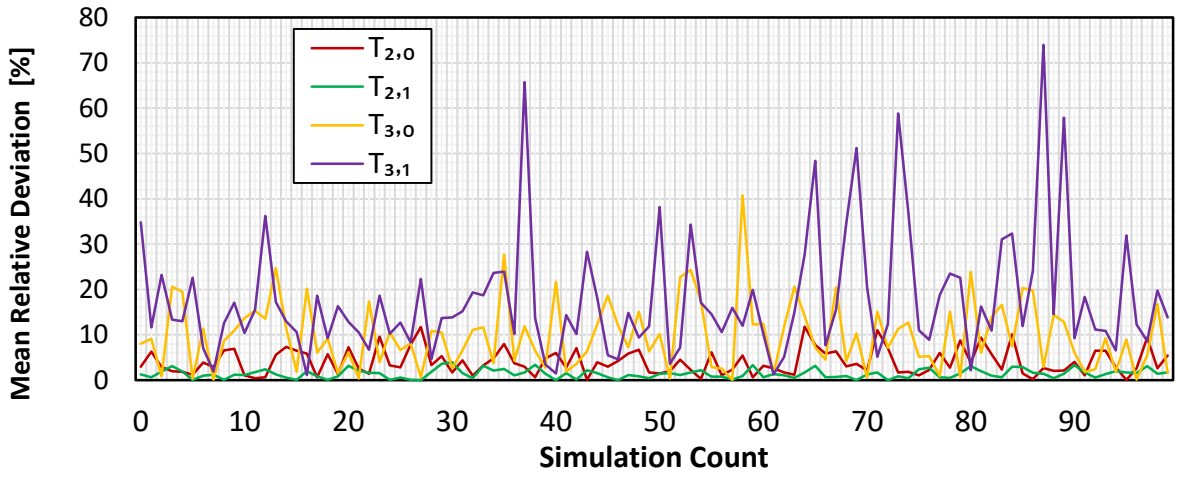


Figure 4.11: Relative Deviation Variation - Second Robustness Testing

	$\hat{T}_{2,0}$	$\hat{T}_{2,1}$	$\hat{T}_{3,0}$	$\hat{T}_{3,1}$
c_R	$\bar{\delta}$			
0	0.13	0.29	1.00	0.81
1	0.22	0.07	0.55	1.00
Average	0.17	0.18	0.7	0.90

Table 4.4: Normalised Mean Relative Deviation of Candidate Models

4.4.4 Decision of Optimal Model

As indicated above, several structures can be used for CPU time modeling. In order to choice an accurate model, it is necessary to define justified criteria to meet the aim of accuracy and robustness. Thus, the choice of the optimal model is then based on the minimization of the cost function defined as

$$\mathcal{C}(\hat{T}_{i,j}) = \bar{\delta}(\hat{T}_{i,j}) + \bar{E}_v(\hat{T}_{i,j}) \tag{4.18}$$

where $\bar{E}_v$ is the normalized mean of the recorded $\bar{e}_v^T$ for all filtering iterations

The validation error is used for evaluation here because it represents the performance of the identified model to the observed data whereas the identification error only indicates the quality of the identification process. Table 4.5 shows the normalized mean error values obtained with straightforward calculations

	$\hat{T}_{2,0}$	$\hat{T}_{2,1}$	$\hat{T}_{3,0}$	$\hat{T}_{3,1}$
$\bar{E}_v$	1.00	0.57	0.64	0.95

Table 4.5: Normalized Mean Validation Error

Clearly, $\hat{T}_{2,1}$ has the corresponding lowest cost function and is followed by $\hat{T}_{3,0}$. The two models are those remarked in the phase of validation. Finally, an accurate model for describing CUP time for Machine A depending on the state size, taking into account the two aspects that are error minimization and the robustness, is given by

$$\hat{T}_{CPU}(n) = 10^{-5}\,(9.55n^2 - 180n^{1.5} + 1380n - 4600n^{0.5} + 5810) \qquad (4.19)$$

The next step is consisting in facing the obtained model with raw data and is consisting of test phase.

4.4.5 Test Phase

In order to illustrate the relevance of the obtained model, the goal of this section is confirming the performances beyond the range used initially for its identification and validation. More precisely, tests are realized over interval $[\![N_s^{max}, N_s^{max} + N_t]\!]$ and data, defined by T_{CPU}, is considered without any filtering where N_t is a time horizon.

A similar simulation is conducted for $n \in [N_s^{max}, N_s^{max} + N_t]$. The identified models for $\hat{T}_{i,j}$ are compared with the observed unfiltered T_{CPU}.

Figure 4.12 depicts results of testing where N_t is chosen to be equal to ten. It should be noted that the structure realizing the best performances during the step of validation, i.e. $\hat{T}_{2,1}$ confirms those performances in the testing phase. Indeed, as it can be checked at the Figure 4.12 or by analyzing the resulting mean relative error that are presented at Table 4.6, the green curve is the closer curve to the blue one representing the observed data.

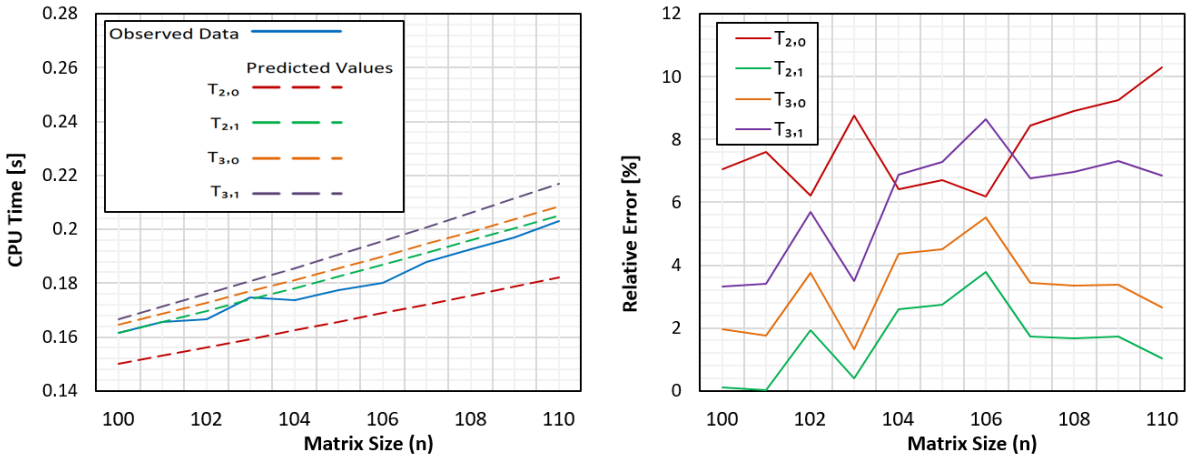


Figure 4.12: Testing Phase of All Identified Models

	$\hat{T}_{2,0}$	$\hat{T}_{2,1}$	$\hat{T}_{3,0}$	$\hat{T}_{3,1}$
$\bar{e}$	7.79	1.61	3.27	6.05

Table 4.6: Mean Relative error during the testing phase

The analysis proposed above allowed obtaining an accurate model for describing the CPU time. For confirming the proposed result, a series of campaigns of tests have been realized in different times. Each campaign was realized in similar environment and provides an equivalent data than one used in the above analysis. The goal is obtaining a global overview on the accuracy based on different observed error results

Table 4.7 presents the recorded relative mean error for all of these tests. The chosen model i.e. $\hat{T}_{2,1}$ offers the best results in all of the tests. In addition, the candidate models are ranked similarly to the tests conducted in Table 4.3 with respect to the validation error and maintain the same ranking across all of the tests shown in Table 4.7 as well. This indicates that the chosen models offers the optimal accuracy even when accounting for variation of matrix size and processor performance which result in variation in the observation data between each test.

	Relative Error (%)			
Test Campaigns (TC)	$\hat{T}_{2,0}$	$\hat{T}_{2,1}$	$\hat{T}_{3,0}$	$\hat{T}_{3,1}$
Identification Test	7.79	1.61	3.27	6.05
TC1	5.87	1.78	4.44	5.04
TC 2	6.44	2.53	5.08	5.49
TC 3	5.63	2.21	4.79	5.37
TC 4	6.07	2.51	4.83	5.30
TC 5	5.65	1.66	4.32	5.01
TC 6	5.90	1.89	4.43	5.06
TC 7	6.02	1.98	4.89	5.26
Average	6.17	2.02	4.50	5.32
Standard Deviation	0.76	0.38	0.59	0.37

Table 4.7: Recorded Mean Error for All Tests

4.4.6 Modelling of solving with reduction methods

This section focuses on providing a model for the calculation time of the *Riccati* process after applying reduction methods, namely *BMR* and *SPA*. As seen in Algorithm II, recall that obtaining G, i.e. (1.3), depends on matrix multiplication of size n. In turn, in the case of *BMR*,

$$A = A_{11}\,, B = B_1\,, C = C_1 \text{ and } D = 0$$

where $A_{11} \in \mathbb{R}^{k\times k}$, $B_1 \in \mathbb{R}^{k\times m}$ and $C_1 \in \mathbb{R}^{p\times k}$

The only set of operations in *BMR* dependent on the reduced matrix size k is value allocation which is insignificant next to matrix multiplication and inversion. Therefore the operations of *BMR* are considered only dependent on n.

However, for *SPA*

$$A = A_{11} - A_{12}A_{22}^{-1}A_{21}\ , B = B_1 - A_{12}A_{22}^{-1}B_2\ , C = C_1 - C_2A_{22}^{-1}A_{21} \text{ and } D = D - C_2A_{22}^{-1}B_2$$

Thus several sets of matrix multiplication and inversion in *SPA* are based on k and cannot be ignored.

When using the reduction method *BMR*, the time calculations is depending on both the original state size, i.e. n, and chosen reduction system state size, i.e. k. The execution total time can be considered as being composed of two independent steps realized successively. The first step is consisting in determining the matrix G while the second step is dedicated to *Riccati* process executed with a reduced state. Hence, the following model can used to express calculation time depending on the two variables n and k.

$$T_{BMR}(n,k) = T_{BMR}^{G}(n) + T_{CPU}(k) \tag{4.20}$$

where T_{BMR}^{G} is the calculation time for obtaining the reduced matrices in the *BMR* process

The model for describing the total time when using the reduction method *SPA* can be established in a similar way. The difference lies in the additional computation (see lines 19 and 20 of Algorithm II) that requires an extra time expressed in the following model by $T_{G,SPA}$.

$$T_{SPA}(n,k) = T_{BMR}^{G}(n) + T_{SPA}^{G}(k) + T_{CPU}(k) \tag{4.21}$$

where T_{SPA}^{G} is the time for obtaining the reduced matrices after obtaining G in the *SPA* process.

4.4.6.1 Identification of Calculation Time

In order to determine explicitly the two models (4.20) and (4.21), the first step is the identification of T_{BMR}^{G} and T_{SPA}^{G} based on the filtered data collected from $n \in [\![N_s^{min}, N_s^{max}]\!]$. Indeed, the *Riccati* process execution model, namely T_{CPU}, is defined by (4.19).

A similar study, as for T_{CPU}, has been realized and similar results have been established for both T_{BMR}^{G} and T_{SPA}^{G} concerning choice of the optimal structures.

Figure 4.13 depicts the identification process and Table 4.8 offers the resulting identification and validation error. One can remark the good accuracy level for both identification and validation. Finally, the established models, for the same machine A, are as follow:

$$\hat{T}_{BMR}^{G}(n) = 10^{-6}(2.58\,n^2 - 14.9\,n^{1.5} + 118\,n - 507\,n^{0.5} + 995) \tag{4.22}$$

and

$$\hat{T}_{SPA}^{G}(k) = 10^{-10}(3018\,k^2 - 2.52\,k^{1.5} - 69479\,k + 71.8\,k^{0.5} + 1.26\,10^6) \tag{4.23}$$

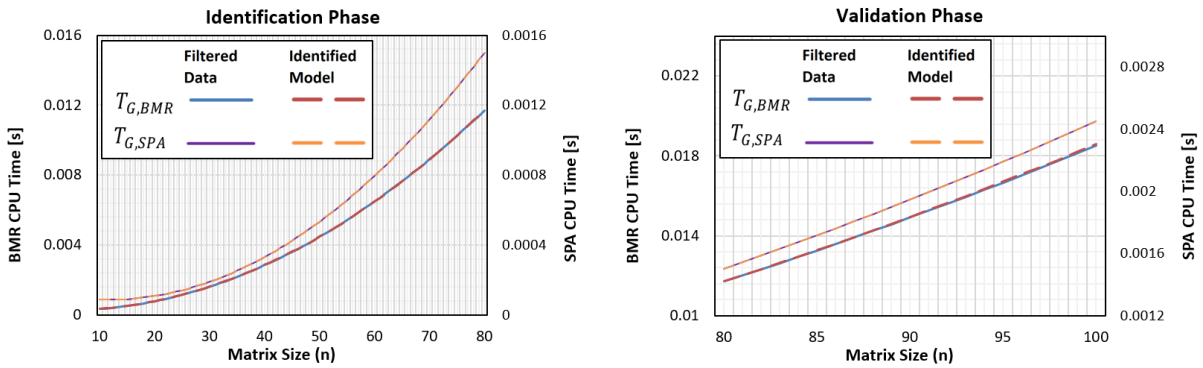


Figure 4.13: Identification for *BMR* and *SPA* Process

	e_i	e_v
$T_{G,BMR}$	0.13	0.23
$T_{G,SPA}$	$2.23\ 10^{-5}$	$4.17\ 10^{-5}$

Table 4.8: Identification and Validation Error for *BMR* and *SPA* Process

Now, based on (1.20) and (1.21), the resulting models for the two approaches are given numerically by

$$\hat{T}_{BMR}(n,k) = 10^{-5}(0.26\,n^2 - 1.5\,n^{1.5} + 11.8\,n - 50.7\,n^{0.5} + 9.5\,k^2 - 180\,k^{1.5} + 1380\,k - 4600\,k^{0.5} + 5909) \quad (4.24)$$

and

$$\hat{T}_{SPA}(n,k) = 10^{-5}(0.26\,n^2 - 1.5\,n^{1.5} + 11.8\,n - 50.7\,n^{0.5} + 9.53\,k^2 - 180\,k^{1.5} + 1379\,k - 4599\,k^{0.5} + 5921.6) \quad (4.25)$$

4.4.6.2 Comparison between Reduction Methods

In this section, a comparison between the employed reduction methods is conducted in order to decide the optimal method for the *Riccati* Process. This is based on two main factors; calculation time and predictability. For the first criteria, *BMR* evidently shows a lower calculation time as seen previously.

For predictability, two main parameters for observation are employed. The first is the Coefficient of Determination, denoted R^2, which is a statistical measure that indicates the proportion of the variance in the dependent variable that is predictable from the independent variable in a model. It measures the goodness of fit of a regression model to the observed data and is considered a measure of predictive efficiency [137]. As for the second, it is the testing error where the identified models are compared to the observed data. The use of the testing error is to assess the adaptation of the data to a mathematical model, in turn evaluating the predictability of the process.

In order to conduct this testing, a similar simulation is conducted where the matrix size n is fixed and the desired reduced matrix size k is variable. Let $n = N_s^{max}$ and $N_s^0, N_s^1 \in [\![N_s^{min}, n]\!]$, with $N_s^0 \leq N_s^1$, be two integers denoting the range used for the state size of the reduced system, i.e. $k \in [\![N_s^0, N_s^1]\!]$.

Figure 4.14 shows results of simulation based the models expressed by (4.24) and (4.25) and experimentation data on machine A. It should be noted the accurate quality obtained by comparing the values obtained from both data and the models.

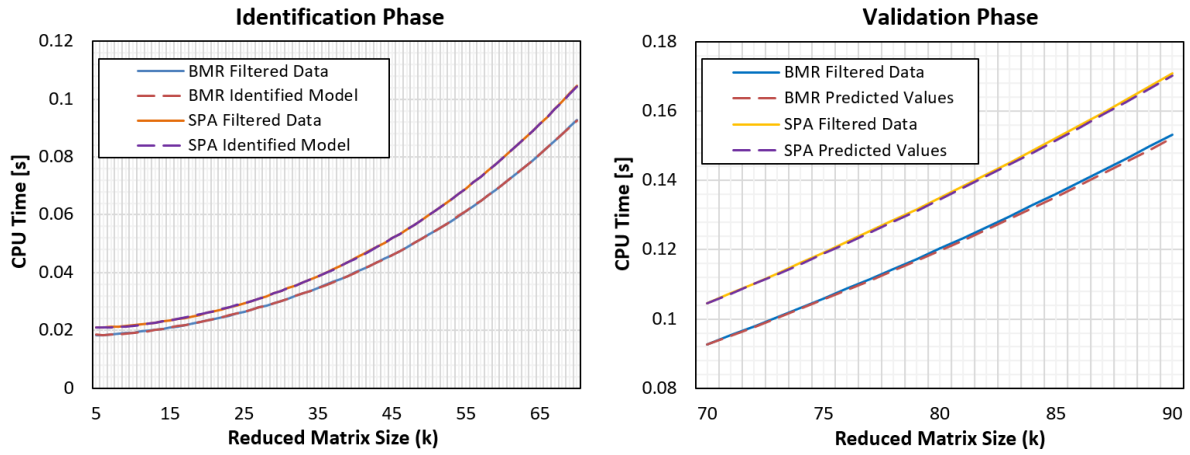


Figure 4.14: Simulation Process with Reduction Methods for Fixed Matrix Size

To compare the considered reduction methods for choosing a relevant one, a coefficient is defined for estimating the approach quality. More precisely, the coefficient of determination is measured between the observed data and the corresponding trend line [138] of the data which, for a given dataset T_{BMR} and T_{SPA}, are defined as

$$\tau_{BMR}(k) = \frac{1}{2W_{MA}+1} \sum_{j=k-w_{MA}}^{k+w_{MA}} T_{BMR}(k+j) \tag{4.26}$$

and

$$\tau_{SPA}(k) = \frac{1}{2W_{MA}+1} \sum_{j=k-w_{MA}}^{k+w_{MA}} T_{SPA}(k+j) \tag{4.27}$$

where w_{MA} is the chosen window for the Moving Average process

Figure 4.15 presents the calculation time with the corresponding trendlines for each reduction method.

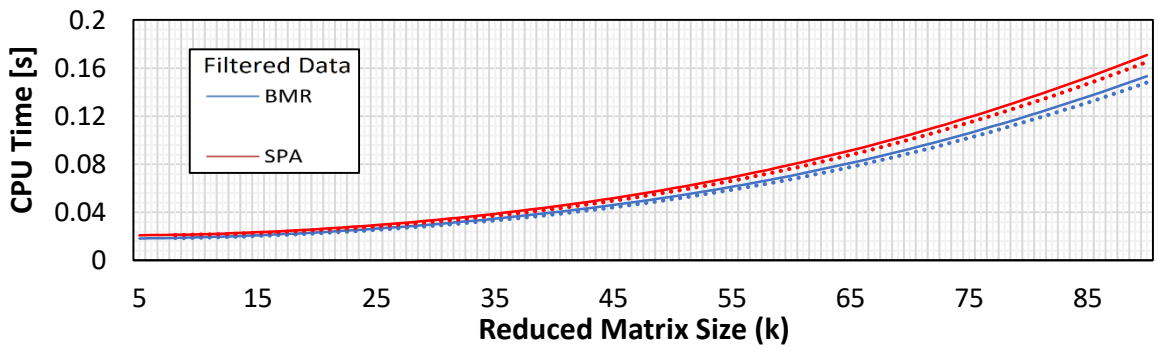


Figure 4.15: Filtered Data of Calculation Time with trend lines

In turn, the coefficient of determination is given as

$$R^2\left(\hat{T}_{\mathfrak{m}}\right) = 1 - \frac{\sum_{k=N_0^S}^{N_1^S}\left(\hat{T}_{\mathfrak{m}}(k) - \tau_{\mathfrak{m}}(k)\right)^2}{\sum_{k=N_0^S}^{N_1^S}\left(\bar{\tau}_{\mathfrak{m}} - \tau_{\mathfrak{m}}(k)\right)^2} \tag{4.28}$$

such that

$$\bar{\tau}_{\mathfrak{m}} = \frac{1}{N_1^S - N_0^S}\sum_{N_0^S}^{N_1^S}\tau_{\mathfrak{m}}(k) \tag{4.29}$$

where $\mathfrak{m} \in \{BMR, SPA\}$

Thus, it is seen that

$$R^2\left(\hat{T}_{BMR}\right) = 1.1\, R^2\left(\hat{T}_{SPA}\right)$$

As for the second parameter of comparison, the error between the identified models and the observed data is studied where the results are shown in Figure 4.16.

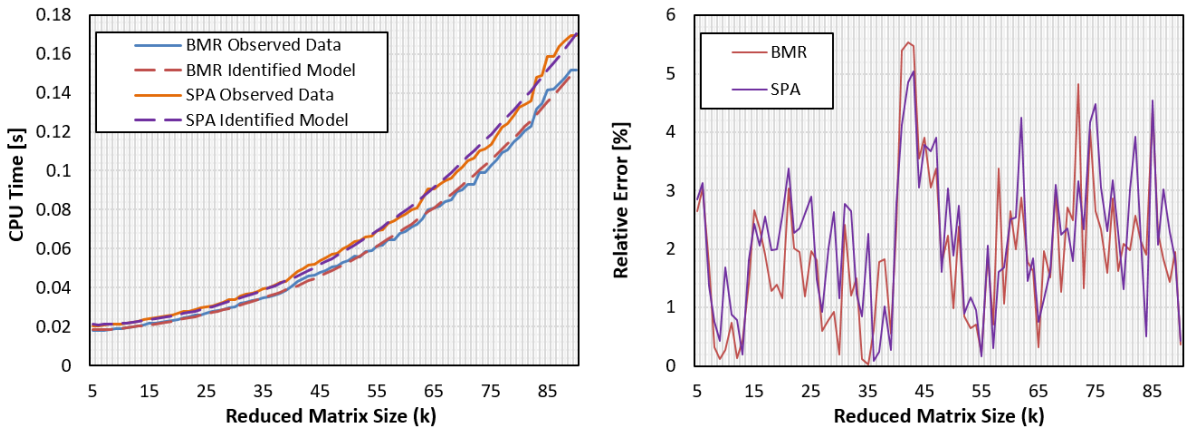


Figure 4.16: Comparison with Observed Data for Reduction Methods

The resulting mean error for the reduction methods is

$$\bar{e}_{BMR} = 1.7\% \quad \text{and} \quad \bar{e}_{SPA} = 2.15\%$$

The higher calculation time of *SPA* and the lower coefficient of determination lead to the choice of *BMR* is the optimal method for the *Riccati* Process in LQR control.

4.5 Model Extrapolation Procedure

This sections focuses on the translation of the identified models for calculation time to other machines. While hardware components would evidently result in different observed results, the objective is to find a common model that accounts for machine characteristics and offers a common set of criteria.

While machines vary in components and characteristics, the main common point for any simulation and calculation is the number of operations. This is fixed for any simulation regardless of the processor it is conducted on. The Floating Point Operations per Second (FLOPS) is standardly used as a metric for computer performance. It acts as a measure of the theoretic peak performance of a given system. However, the measurement of FLOPS has become more complex with modern technology. Before, multiplying the number of floating-point operations per cycle by the number of cycles per second was sufficient. System advancement such as dynamic clock speed, hyperthreading and parallel computing render this measurement unfeasible by simple calculation [139].

4.5.1 Procedure

With the number of operations as the common variable between all machines, the aim is to define a model for this number which can be used for all simulation platforms. While theoretical definitions of number of operations for a task can be deducted such as in the case of matrix multiplication, this process is infeasible in the case of the *Riccati* process and model reduction methods because of the use of the *Riccati* solution, Lyapunov process and Cholesky decomposition. These tasks rely on a set of iterations until convergence of the output which in turn removes the ability of defining a model for the number of operations. Thus this process is based on the previously defined calculation time models alongside the study of FLOPS.

The first step is the determination of the Machine A's FLOPS. There are diverse sets of application and third party tools that offer the data required for this approximation. In summary, it offers a benchmark test that records the number of operations during a given timeframe. In this work, this test is achieved through Matlab. The observed simulation is matrix multiplication because of its known number of operations required. The simulation is repeated for a large set of instances and FLOPS can be deduced by dividing the number of operations for matrix multiplication by the average calculation time. Considering $A\,B$ the symbol of matrix multiplication and $\hat{O}(A\,B)$ the estimated number of operations associated to it, then

$$FLOPS_A = \frac{\hat{O}(A\,B)}{\bar{T}(A\,B)} \tag{4.30}$$

where $\bar{T}(A\,B)$ is the mean of the calculation time for the execution of the associated matrix multiplication

For this simulation, $A, B \in \mathbb{R}^{n\times n}$, $n \in [\![N_S^{min}, N_S^{max}]\!]$ and $\hat{O}(A\,B) = n^3$ (see for instance [140]). The simulation is repeated for N^{FLOPS} times per n and $\bar{T}_A(A\,B)$ is the mean of the calculation times recorded per simulation on Machine A.

Figure 16 shows a sample of the conducted simulations where $n = N_s^{max}$. The calculation time average and deviation denoted ρ_A are

$$\bar{T}_A(A\,B) = 6.68\,10^{-5}s \qquad \text{and} \qquad \boldsymbol{\sigma}_A(A\,B) = 9.66\,10^{-6}$$

The recorded standard deviation alongside the variations presented in Figure 4.16 indicate the lack of guaranty of perfect repeatability with sequential machine performance and reassures the need for large sets of simulations for mean calculation as adopted in this entire work.

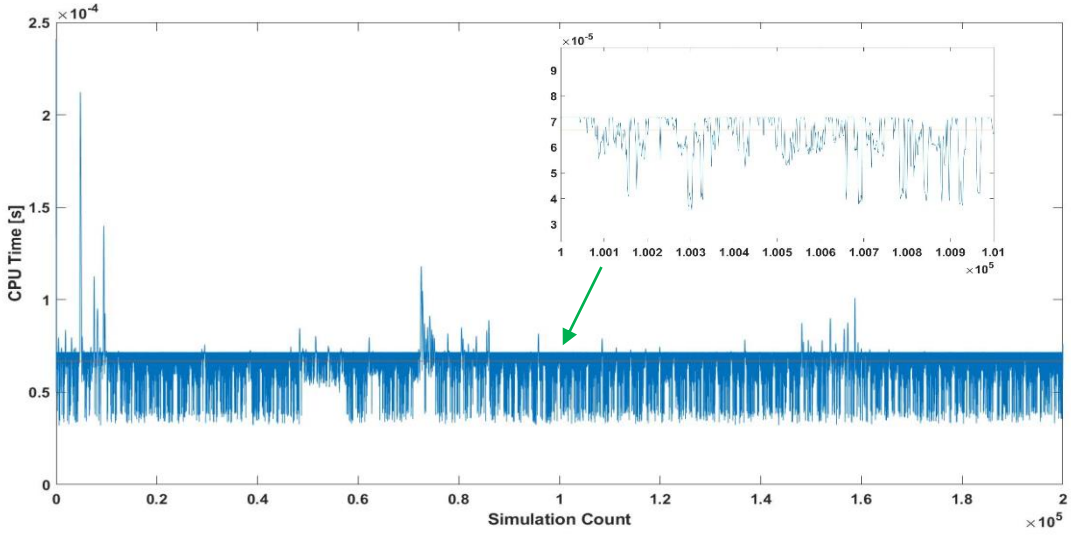


Figure 4.17: FLOPS calculation time sample for Machine A

The resulting FLOPS for Machine A is

$$FLOPS_A = 1.415\,10^{10}\ operations/second$$

Based on this value, the model for the number of operations can be defined as

$$\hat{O}_A = FLOPS_A\,\hat{T}_A$$

$$\hat{O}_B = FLOPS_B\,\hat{T}_B$$

where $\hat{T}_A, \hat{T}_B$ is the calculation time for a given operation in Machine A and Machine B respectively.

Given as $\hat{O}_1$ and $\hat{O}_2$ are similar for all machines, the next step is defining the calculation time model for a specific processor based on the general number of operations. Thus, given a machine x, the resulting time for calculation is as follows

$$\hat{T}_{x,CPU} = \frac{FLOPS_A}{FLOPS_x}\,\hat{T}_{CPU} \tag{4.31}$$

And in turn

$$\hat{T}_{x,CPU}(n) = \frac{10^5}{FLOPS_x}(13.51\,n^2 - 254.7\,n^{1.5} + 1952.7\,n - 6500\,n^{0.5} + 8220) \tag{4.32}$$

4.5.2 Validation Test

In order to verify the model presented in (4.30), the *Riccati* simulation is conducted on Machine B i.e. T_{CPU}. The objective is to study the error between the identified model in (4.17) and the observed simulation result. The first step is to identify FLOPS of the targeted machine. Figure 4.17 shows the same FLOPS test sample for Machine B and the calculation time average and deviation are respectively found as

$$\bar{T}_B(A\,B) = 2.17\,10^{-4}s \qquad \text{and} \qquad \boldsymbol{\sigma}_B(A\,B) = 1.85\,10^{-5}$$

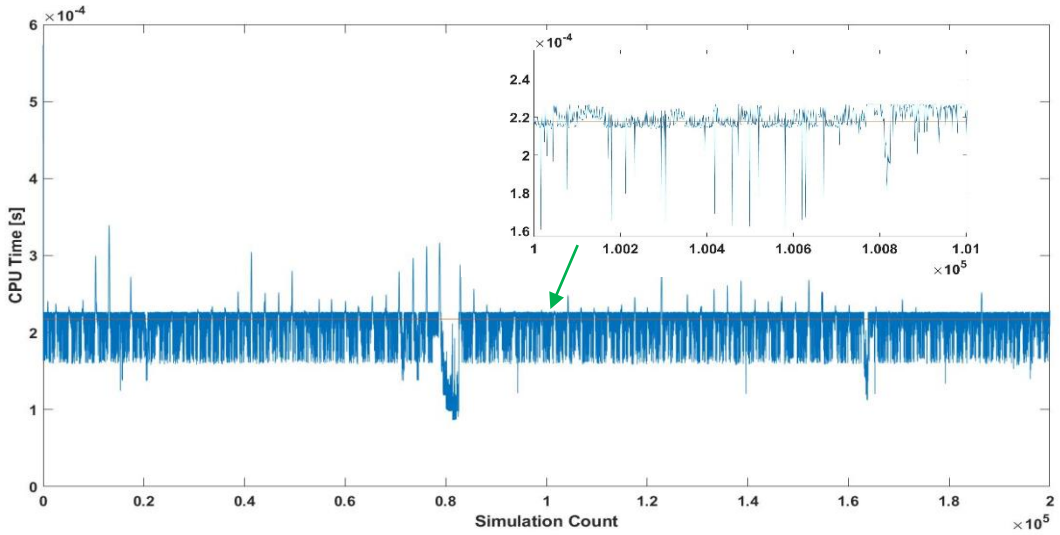


Figure 4.18: FLOPS calculation time sample for Machine B

Machine B results in a larger calculation time as expected from the specifications offered in Table 4.1 alongside similar variations in calculation time.

The resulting FLOPS for the studied machine is

$$FLOPS_B = 2.36\,10^9\ operations/second$$

Afterwards, the simulation is conducted and the identification process is executed between the observed simulation and the defined model.

The relative error average is calculated based on (4.5) and the mean is equal to 2%,

Figure 4.18 depicts the identified model and resulting relative error and corresponding Trend Line for the *Riccati* calculation time of Machine B. The corresponding trend line shows the decrease of the relative error for larger matrix sizes. This is primarily because of the smaller calculation times for lower values of the variable *n* which can lead to perturbations in the accuracy of the observations.

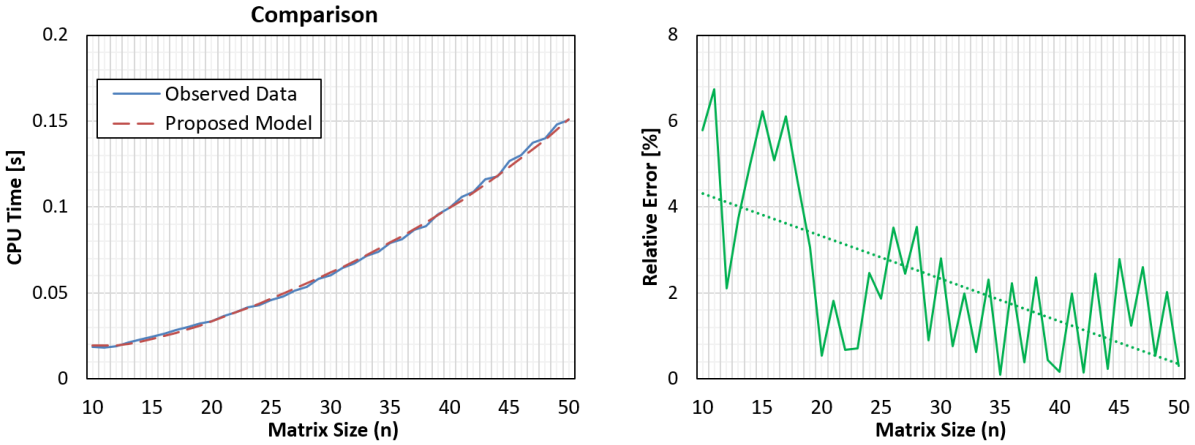


Figure 4.19: Identification of *Riccati* Process on second machine

4.6 Analysis of Precision Loss Vs CPU Time Gain

The setup of the models for each procedure allows the prediction of the required order of k to obtain a desired CPU Time reduction. However, this reduction comes at the cost of increasing approximation level and in turn approximation error. The objective of this section is the setup of a cost function that offers a quantitative description of the relation between the calculation time and the precision variation. The reduction error in study here is based on the Hankel Singular Values. The interest in this error is because the calculation of the singular values and the error bound later on is based entirely on the original state-space matrices in concern and doesn't necessitate the resulting reduced matrices. This is primarily because certain types of reduction possess properties which allow the interpretation of the error (see [141]) and that is the case for *BMR* as well. This offers a good starting point in the work in order to reach the main objective of approximating time gain and precision loss in the *Riccati* Process after reduction without the need for any computationally expensive operations.

A case study in the field of Adaptive Optics is studied here to show the use of the cost function. The singular values are employed because of its standard use for capturing the quality of approximation of transfer functions by stable smaller transfer functions of McMillan degree [142].

4.6.1 Setup of the relationship cost function

Considering Algorithm II such that $P_1 = L_1 L_1^T$ and $P_2 = L_2 L_2^T$ which are the observability and controllability gramians, and obtaining

$$\begin{cases} AP_1 + P_1 A^T + BB^T = 0 \\ A^T P_2 + P_2 A + C^T C = 0 \end{cases} \tag{4.33}$$

For Singular Value Decomposition based methods i.e. *BMR* and *SPA*, the Hankel Singular values σ [143] are defined as

$$\sigma_i = \sqrt{\lambda_i(P_1\,P_2)}\ , \qquad i \in [\![1, n]\!] \tag{4.34}$$

where λ_i are the matrix eigenvalues

The H_∞ norm is a measure used in control theory and signal processing to quantify the worst-case gain of a system over all frequencies. In simpler terms, it represents the maximum amplification factor that the system can impart on any input signal, across all frequencies. In the case of this work where G and G_k represent the transfer functions of the original and reduced system respectively, $\|G - G_k\|_\infty$ provides a measure of the approximation quality in the worst-case scenario [144] with

$$\|\boldsymbol{G}\|_\infty = \sigma_{max}(\boldsymbol{G}(j\omega)) \quad \text{for all} \quad \boldsymbol{G} \in \mathbb{R}^{(n+p)\times(n+m)}$$

where σ_{max} is the largest singular value

In the case of *BMR*, it provides a computable error bound [141] in terms of σ such that

$$\|G - G_k\|_\infty \leq 2 \sum_{i=k+1}^{n} \sigma_i \tag{4.35}$$

The objective of the presented cost function is to present a quantitative analysis of the efficiency of the reduction process based on gain in calculation time and loss in precision based on the concerned error bound and is defined as

$$\mathcal{C}(n,k) = \mathcal{C}_{CPU}(n,k) + \mathcal{C}_\infty(n,k) \tag{4.36}$$

where, in the case of *BMR*

$$\mathcal{C}_{CPU}(n,k) = \frac{T_{BMR}(n,k)}{T_{CPU}(n)} \tag{4.37}$$

and

$$\mathcal{C}_\infty(n,k) = \frac{\sum_{i=k+1}^{n} \sigma_i}{\sum_{i=1}^{n} \sigma_i} \tag{4.38}$$

Note that the two functions are normalized in order to progress towards the global optima of the cost function (see [136]).

4.6.2 Decay Rate Approximation

While the previously proposed cost function offers a method for balancing between time gain and precision loss, one of its major setbacks is its reliance on the calculation of the Hankel Singular values in order to assess F_∞. In a similar approach to calculation time approximation, the interest is to provide the model for the evolution of the Hankel singular values without their direct calculation. The interest in this approximation is because of the complexity behind calculating the observability and controllability gramians which are needed for the Hankel

Singular Values. These gramians can be computationally expensive to find and analyze for non-linear dynamic systems especially for large matrix sizes [145].

A possible alternative route for the evaluation between the calculation time gain and the precision is the assessment of the decay rate of σ instead of their specific values. Because of this, the work here focuses on the replacement of the calculation of σ with its decay rate variation in the set cost function in particularly for SISO systems.

In order to achieve this, let the transfer function of (4.1) be

$$G(s) = C(sI - A)^{-1}B + D = \frac{p(s)}{q(s)} \tag{4.39}$$

And γ is defined as

$$\gamma_i = \frac{p(\lambda_i(A))}{q^*(-\lambda_i(A))} \ , \ i \in [\![1, n]\!] \tag{4.40}$$

where q^* is the conjugate of q

Then, based on [141], the following inequality is established

$$\sum_{i=1}^{k} \log|\gamma_{n-i+1}| \geq \sum_{i=1}^{k} \log(\sigma_{n-i+1}) \quad \text{for } j \in [\![1, n]\!] \tag{4.41}$$

in turn

$$\sum_{i=k}^{n} \log|\gamma_i| \geq \sum_{i=k}^{n} \log(\sigma_i) \tag{4.42}$$

Therefore, a new version of $\mathcal{C}_\infty$ can be modeled based on γ such as

$$\mathcal{C}_{\infty,\gamma}(n, k) = \frac{\sum_{i=k}^{n} \log|\gamma_i|}{\sum_{i=1}^{n} \log|\gamma_i|} \tag{4.43}$$

and

$$\mathcal{C}(n, k) = \mathcal{C}_{CPU}(n, k) + \mathcal{C}_{\infty,\gamma}(n, k) \tag{4.44}$$

4.6.2.1 Simulation for Comparison

For this simulation, let $n = N_S^{max}$. A sample of N_S^A simulations is conducted where the parameters of the system per simulation are randomized. The cost function is calculated in each where the first is with $\mathcal{C}_\infty$ and the second with $\mathcal{C}_{\infty,\gamma}$. Since this is an optimization problem, the value of the desired reduced matrix size that minimizes the cost function and indicates the global optima is observed in both cases and compared through the approximation relative error defined as

$$e_A(i) = \frac{|k_\infty - k_{\infty,\gamma}|}{N_S^{max}} \tag{4.45}$$

where k_∞ and $k_{\infty,\gamma}$ are the k values to minimize $\mathcal{C}$ based on using $\mathcal{C}_\infty$ and $\mathcal{C}_{\infty,\gamma}$ respectively and $i \in [\![1, N_s^e]\!]$ is the index of the simulation number and $N_s^{e_A}$ is the maximum number of simulations conducted

Figure 4.19 presents the approximation relative error from the simulation where its mean is equal to

$$\bar{e}_A = \frac{1}{N_s^e} \sum_{i=1}^{N_s^e} e_A(i) = 6\%$$

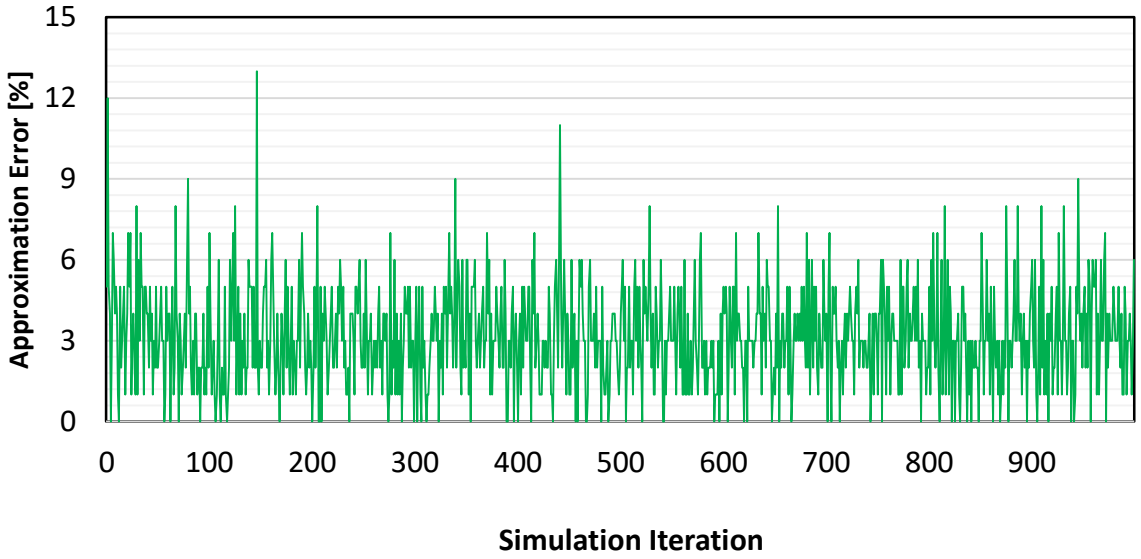


Figure 4.20: Cost Function Approximation Relative Error

4.6.2.2 Analysis of resulting Error

To study the resulting difference between the minimum indices found in Figure 4.19, the variation between $\mathcal{C}_{\infty,\gamma}$ and $\mathcal{C}_\infty$ for the past simulation is studied and the results are in Figure 4.20.

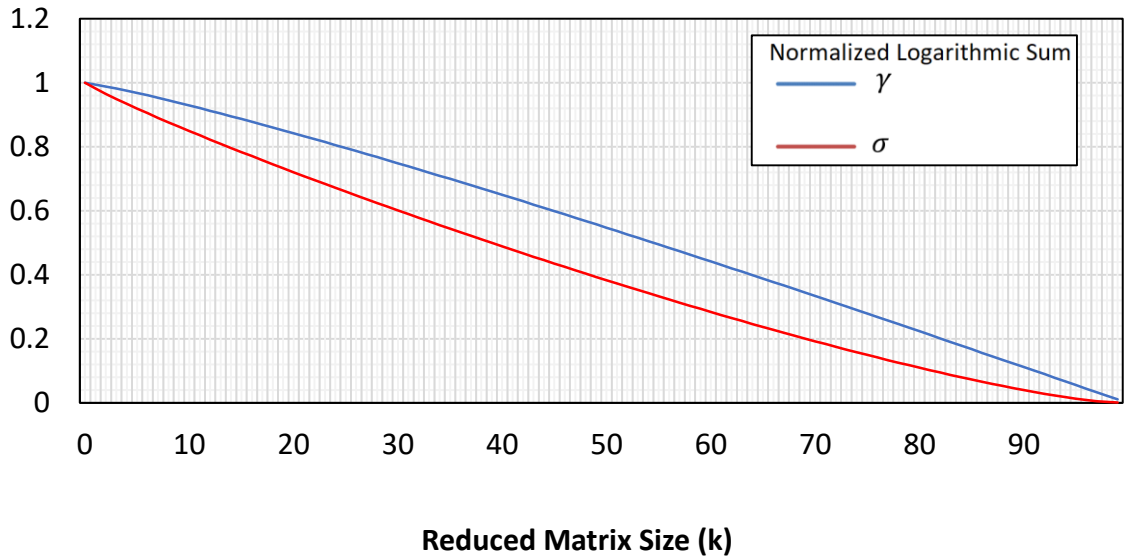


Figure 4.21: Normalized Logarithmic Sums

Based on the results of the previous simulation, a sharper upper boundary of the logarithmic sum of the Hankel Singular values is sought after. Considering $\mathcal{L}$ as the line defined as

$$\mathcal{L}(x) = \frac{n-x}{n-1}$$

It should be noted that the two points $(1,1)$ and $(n,0)$ are on the line $\mathcal{L}$. Let $\mathcal{C}_{\mathcal{L}}$ represent the symmetry of $\mathcal{C}_{\infty,\gamma}$ with respect to $\mathcal{L}$. Let $\{p_1, \dots, p_\emptyset\} \in \mathcal{C}_{\infty,\gamma}$, $\{p_1', \dots, p_\emptyset'\} \in \mathcal{L}$ and $\{p_1'', \dots, p_\emptyset''\} \in \mathcal{C}_{\mathcal{L}}$ be the set of points with $(x_i, y_i), (x_i', y_i'), (x_i'', y_i'')$ their respective coordinates for $i \in [\![1, \phi]\!]$ where $\phi \in \mathbb{N}^*$ denotes the number of points sampled in each group. Then the symmetric function is defined by

$$p_i'' : (2x_i' - x_i\,, 2y_i' - y_i)$$

Figure 4.21 shows the corresponding $\mathcal{C}_\infty, \mathcal{C}_{\mathcal{L}}$ and $\mathcal{C}_{\infty,\gamma}$.

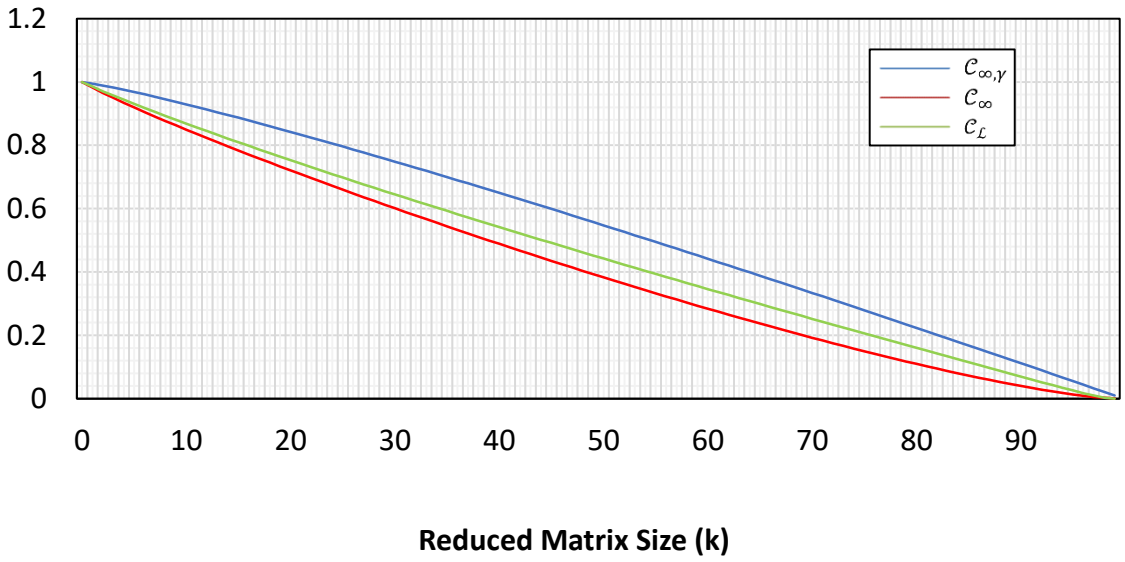


Figure 4.22: Logarithmic sum and approximations of simulations

Let ϑ indicate the new approximation and is given by

$$\vartheta(n,k) = \frac{2(n-k)}{n-1} - \frac{\sum_{i=k}^{n} \log|\gamma_i|}{\sum_{i=1}^{n} \log|\gamma_i|} \tag{4.46}$$

and in turn

$$\mathcal{C}_\vartheta(n,k) = \mathcal{C}_{CPU}(n,k) + \vartheta(n,k) \tag{4.47}$$

Simulations similar to Figure 4.19 are conducted with the new approximation cost function where the resulting error is now $\bar{e}_A = 3\%$. The reduction in the error indicates the potential in using the approximation in (4.46) as a viable alternative for the approximation of the decay rate of Hankel Singular Values without directly calculating them.

4.7 Case Study

This section presents the application of the defined models and resulting cost function in a real-time domain. After the setup of the control model and the corresponding parameters, the relationship between the reduction in calculation time after *B*alanced *M*odel *R*eduction and the resulting reduction error is studied. The interest in the domain of Adaptive Optics is because of the use of the Discrete Algebraic *Riccati* Equation for Kalman Filtering which can be quite resource consuming especially for modern generations of telescopes [146].

4.7.1 Concept of Adaptive Optics Control

Adaptive optics (AO) is a technology used in optics, particularly in telescopes and microscopes, to compensate for the distortion caused by the Earth's atmosphere or other optical aberrations. It works by continuously measuring the distortions in the incoming wavefront (WF) caused by atmospheric turbulence or imperfections in optical systems, and then rapidly adjusting the shape of a deformable mirror or other optical elements to correct for these distortions in real-time. This enables astronomers to obtain clearer images of celestial objects by reducing the blurring effects of atmospheric turbulence, and it also enhances the resolution of microscopes by compensating for optical aberrations. Adaptive optics has revolutionized astronomical observations, allowing astronomers to see more detail in distant objects, and it has applications in various fields including laser communication, laser weaponry, and medical imaging [147]. In this section, the focus is on Multiconjugate AO (MCAO). Figure 4.22 offers an overview of the Adaptive Optics concept.

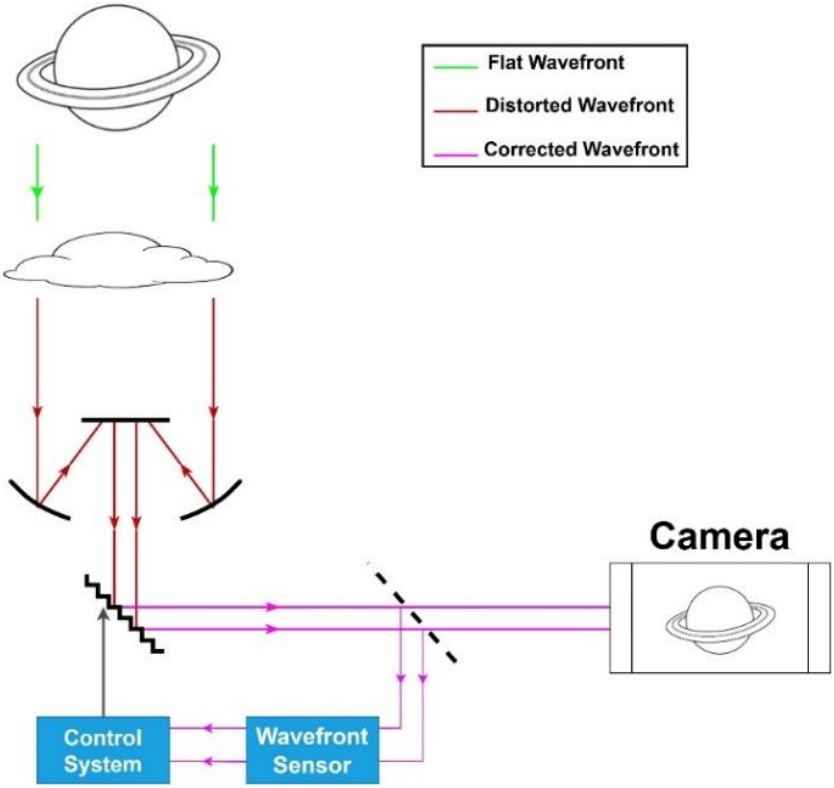


Figure 4.23: Adaptive Optics Process

4.7.2 Closed-Loop Control Process

Our interest in this domain is in the closed-loop correction of the turbulent phase, based on [148]. Figure 4.23 shows the control model.

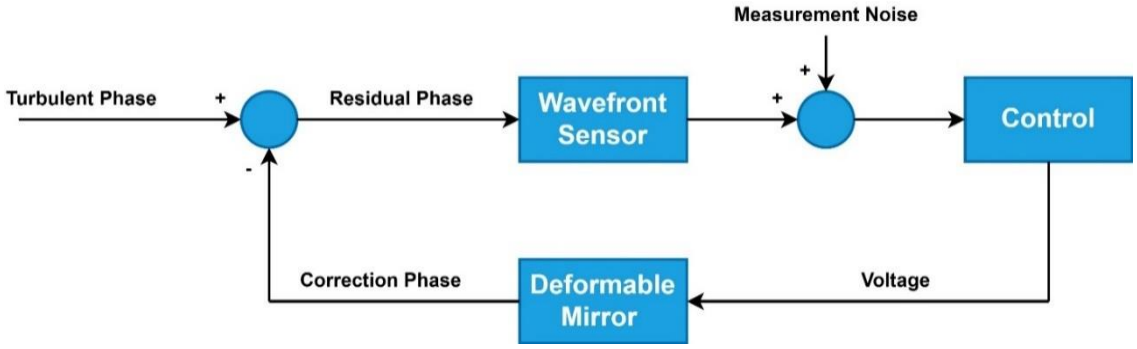


Figure 4.24: Adaptive Optics Turbulent Phase Closed-Loop

Let $\emptyset^{tur}, \emptyset^{cor}, \emptyset^{res} \in \mathbb{R}^{q_1}$ be the turbulent, correction and residual phases respectively such that

$$\emptyset^{res}(T) = \emptyset^{tur}(T) - \emptyset^{cor}(T) \tag{4.48}$$

where $\mathcal{T} \in \mathbb{N}^*$ is the sampling period for observation

The resulting phases $\emptyset$ in a given direction d_j are the sum of the resulting phase at each layer l, noted $\varphi_{l,i}$ on interval $[(i-1)\mathcal{T}, i\mathcal{T}]$ $(i \in \mathbb{N}^*)$, then

$$\emptyset^{tur}_{\mathrm{d}_j,i} = M^L_{\mathrm{d}_j}\varphi^{tur}_i \tag{4.49}$$

and

$$\emptyset^{cor}_{\mathrm{d}_j,i} = M^{DM}_{\mathrm{d}_j}\varphi^{cor}_i \tag{4.50}$$

such that

$$\varphi^{tur}_i = \left[(\varphi^{tur}_{1,i})^T, (\varphi^{tur}_{2,i})^T, \dots, (\varphi^{tur}_{L,i})^T\right]^T \tag{4.51}$$

Therefore for all directions d, the equations are equivalent to

$$\emptyset^{tur}_{\mathrm{d},i} = M^L_{\mathrm{d}}\varphi^{tur}_i \tag{4.52}$$

and

$$\emptyset^{cor}_{\mathrm{d},i} = M^{DM}_{\mathrm{d}}\varphi^{cor}_i \tag{4.53}$$

where $M^L_{\mathrm{d}} \in \mathbb{R}^{q_1 \times q_2}, M^{DM}_{\mathrm{d}} \in \mathbb{R}^{q_3 \times q_4}$ and L is the total number of turbulent layers that are used to model the atmosphere

The variation of the turbulent phase at layer l is studied as an Autoregressive model defined as

$$\varphi^{tur}_{l,i+1} = A_{r,l}\varphi^{tur}_{l,i} + w_i \tag{4.54}$$

$A_{r,l} \in \mathbb{R}^{q_5 \times q_5}$ is standardly chosen as a diagonal matrix with the value of each diagonal element [149] is defined as

$$a_{r,l}^{jj} = e^{\frac{-0.3V\mathcal{T}(r_O(j)+1)}{\mathcal{D}}} \tag{4.55}$$

where r_O is the radial order of the Zernike mode i, V is the wind speed in the Turbulent layer and $\mathcal{D}$ is the pupil diameter

The overall turbulence is then defined as

$$\varphi_{i+1}^{tur} = A_r \varphi_i^{tur} + w_i \tag{4.56}$$

where $A_r \in \mathbb{R}^{q_6 \times q_6}$ is a block diagonal composed of the diagonal concatenation of $A_{r,l}$

$\varphi_{l,i}^{tur}$ is represented by a vector of Zernike coefficients [150] and $\varphi_i^{tur} \in \mathbb{R}^{q_7}$ is represented by the concatenation of the vectors of each layer.

Zernike coefficients [150], for a given polar coordinate $\boldsymbol{r} = (r, \theta)$, are defined as

$$Z_i = \begin{cases} \sqrt{r_0 + 1}\, R_{r_0}^{r_\theta} \sqrt{2} \cos(r_\theta \theta) & \text{if } r_\theta \neq 0 \text{ and } i \text{ is even} \\ \sqrt{r_0 + 1}\, R_{r_0}^{r_\theta} \sqrt{2} \sin(r_\theta \theta) & \text{if } r_\theta \neq 0 \text{ and } i \text{ is odd} \\ \sqrt{r_0 + 1}\, R_{r_0}^{r_\theta} & \text{if } r_\theta = 0 \end{cases} \tag{4.57}$$

such that

$$R_{r_0}^{r_\theta}(r) = \sum_{j=0}^{\frac{r_0 - r_\theta}{2}} \frac{(-1)^j (r_0 - j)!}{j! \left(\frac{r_0 + r_\theta}{2} - j\right)! \left(\frac{r_0 - r_\theta}{2} - j\right)!} r^{r_0 - 2j} \tag{4.58}$$

where r_0 and r_θ are the radial order and azimuthal order respectively and i is the order of the Zernike mode

The points $\boldsymbol{r}$ are based on the choice of Guide Stars (GS). AO systems require a wavefront reference source of light i.e. GS. Natural stars can serve as point sources for this purpose. Let T_{GS} be the total number of GS used.

The Zernike polynomials form a series of continuous and orthogonal polynomials defined across a unit circle. They serve as a mathematical model for describing optical wavefronts as they propagate through various systems. Compared to other sequences, Zernike polynomials present unique benefits. The turbulent phase is described here by its Zernike polynomial expansion. Employing the normalized Zernike expansion for aberration description provides the advantage of each mode's coefficient directly indicating the root mean square wavefront error associated with that mode [151]. The number of Zernike modes presented is dependent on the maximum radial order r_{max} presented such that

$$r_{modes} = \sum_{j=1}^{r_{max}} (j + 1) \tag{4.59}$$

Then, based on Gaussian Summation

$$r_{modes} = \frac{r_{max}(r_{max}+1)}{2} + r_{max} = 0.5r_{max}(r_{max}+3) \tag{4.60}$$

The WF Sensor is considered linear and integrates the input signal over the specified time period where the measurement signal is defined as

$$y_i = 2\emptyset_{i-1}^{res} + w_{i-1} \tag{4.61}$$

where w is a white Gaussian noise and $2 \in \mathbb{R}^{q_8 \times q_9}$ describes the linear response of the WF Sensor

The specific dimensions of Z_{WF} vary depending on the type of sensor used. One of the common sensor types employed in this field is the Shack-Hartmannn sensor where the output signal is the gradient of the phase sampled at the center of the squares delimited by four sample points of the phase. Considering the phase at these four corners as $\emptyset_a, \emptyset_b, \emptyset_c, \emptyset_d$, then Z_{WF} is the matrix that expresses the gradient formula s for all the points in the pupil where, at a given point observed on the sensor

$$s = \frac{1}{2}\begin{bmatrix}(\emptyset_b + \emptyset_d) - (\emptyset_a + \emptyset_c)\\(\emptyset_a + \emptyset_b) - (\emptyset_c + \emptyset_d)\end{bmatrix} \tag{4.62}$$

The Deformable Mirror is also assumed to have a linear behavior resulting in

$$\emptyset_{i+1}^{cor} = Fv_i \tag{4.63}$$

where $F \in \mathbb{R}^{q_{10} \times q_{11}}$ and $v_i \in \mathbb{R}^{q_{12}}$ is the control voltage

4.7.3 Matrix Dimension Specifications

Based on the previous description, it is evident that the size of the matrices of the state-space model that represents a MCAO system vary based on several parameters. The aim here is to determine $q_1 \dots q_{12}$ based on the physical specifications of the system.

Based on φ_i^{tur}, it is represented by Zernike coefficients per layer per GS taken where $\mathfrak{a}_{GS}$ denotes the total number of GS taken, therefore

$$q_7 = q_2 = q_4 = q_6 = q_\varphi = 0.5L\mathfrak{a}_{GS}\, r_{max}(r_{max}+3)$$

and

$$q_5 = \frac{q_\varphi}{L}$$

$M_{\mathfrak{d}}^{L} \in \mathbb{R}^{q_1 \times q_2}$ *and* $M_{\mathfrak{d}}^{DM} \in \mathbb{R}^{q_3 \times q_4}$ represent the sum of the phase in all directions, therefore for $\mathfrak{a}_{\mathfrak{d}}$ as the total number of directions taken, then

$$q_1 = q_3 = \mathfrak{a}_{\mathfrak{d}}$$

The observations by the WFS are based on the number of points taken denoted $\mathfrak{a}_{WFS}$, then

$$q_8 = T_{WFS} \text{ and } q_9 = \mathfrak{a}_{WFS}$$

For the control voltage, let $\mathfrak{a}_v$ be the number of control voltages observed, giving

$$q_{11} = q_{12} = \mathfrak{a}_v \quad \text{and} \quad q_{10} = \mathfrak{a}_{\mathfrak{d}}$$

4.7.4 State-Space Control Model

The state-space model [152] is defined as

$$X_{i+1} = \begin{bmatrix} A_r & \mathbf{0}_{q_\varphi} & \mathbf{0}_{q_\varphi} & \mathbf{0}_{q_\varphi} \\ I_{q_\varphi} & \mathbf{0}_{q_\varphi} & \mathbf{0}_{q_\varphi} & \mathbf{0}_{q_\varphi} \\ \mathbf{0}_{q_\varphi} & \mathbf{0}_{q_\varphi} & \mathbf{0}_{q_\varphi} & \mathbf{0}_{q_\varphi} \\ \mathbf{0}_{q_\varphi} & \mathbf{0}_{q_\varphi} & I_{q_\varphi} & \mathbf{0}_{q_\varphi} \end{bmatrix} X_i + \begin{bmatrix} \mathbf{0}_{q_\varphi} \\ \mathbf{0}_{q_\varphi} \\ I_{q_\varphi} \\ \mathbf{0}_{q_\varphi} \end{bmatrix} v_i + \begin{pmatrix} v_i \\ 0_{q_\varphi} \\ 0_{q_\varphi} \\ 0_{q_\varphi} \end{pmatrix} \tag{4.64}$$

$$Y_i = Z_{WF}\left[\mathbf{0}_{q_\varphi} \quad \mathbf{0}_{q_\varphi} \quad M_\beta^L \quad \mathbf{0}_{q_\varphi} \quad -M_\beta^{DM} F\right] X_i + w_i \tag{4.65}$$

where

$$X_{i+1} = \begin{pmatrix} \varphi_{i+1}^{tur} \\ \varphi_i^{tur} \\ v_i \\ v_{i-1} \end{pmatrix}$$

Note that $\mathfrak{a}_v = \mathfrak{a}_{\mathfrak{d}} = q_\varphi$ for the closed loop control process in order to satisfy (4.64).

Concerning the control process, the employed controller is a LQG controller, as defined in [146], where the objective is to find the Kalman gain K where

$$K = APC^T(CPC^T + \Sigma_w)^{-1} \tag{4.66}$$

Where P is the solution for the corresponding *Riccati* equation for the LQG process defined as

$$P = APA^T + \Sigma_v - APC^T(CPC^T + \Sigma_w)^{-1}CPA^T \tag{4.67}$$

where Σ_v and Σ_w are the covariance matrices of v and w respectively

Considering $A, P, \Sigma_v \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{p\times n}$ and $\Sigma_w \in \mathbb{R}^{m\times m}$ and in the case of the AO model where

$$n = 5\, q_\varphi \quad \text{and} \quad m = p = q_\varphi$$

Then the *Riccati* equation for the LQG process for AO is found equivalent to (1.2) in number of operations and in turn calculation time per iteration. In addition, as highlighted in [125], the iterations required for convergence are not affected by the control type of the system i.e. LQR or LQG. Therefore the equivalence is both per iteration and for the entire process. In turn, the same calculation model that was previously defined is applied for both processes. The calculation time model for the *Riccati* Process in the case of MCAO is then found with matrix size equal to

$$n = 2\, L\mathfrak{a}_{GS} r_{max}(r_{max} + 3) \tag{4.68}$$

4.7.4.1 Simulation Process

The simulation conditions taken here are inspired by [152]. They are representative of near-infrared observation on a 8-m telescope at a good astronomical site For simplicity, the number of GS observed and the number of layers are set to one. The wind speed to diameter ratio is set at $\frac{V}{D} = 2\text{Hz}$ and the sampling period is $\mathcal{T} = 0.01s$. The WF Sensor directly measures the Zernike polynomials and the DM directly corrects them, meaning that the matrix Z and F are assumed to be identity. M_β^L and M_β^{DM} are also assumed to be identity.

4.7.4.1.1 Calculation Time model

The variation of several parameters in AO can directly impact the precision of the control process. In this work, the primary interest is in the Zernike radial order. Higher-order Zernike polynomials represent more complex and localized aberrations. By including higher-order terms, the AO system can model and correct finer details of the wavefront distortions, improving the precision of the correction. This simulation focuses on testing the validity of the proposed model with respect to the variation of r_{max} where, for each value, a set of N_S^A simulations is conducted and their mean is recorded. These observations are then compared to the proposed calculation time model which is recalled

$$\hat{T}_{CPU}(n) = 10^{-5}\,(9.55\,n^2 - 180\,n^{1.5} + 1380\,n - 4600\,n^{0.5} + 5810)$$

where, in the case of AO, n is obtained in (4.68)

The interest in this is offering a direct relation between the calculation time and the precision of the control process. The sampled values of the matrix size represents the resulting physical parameters of the system based on chosen Zernike radial orders. Figure 4.25 shows this testing phase where the mean relative error is equal to 2%. This indicates a consistency in the results of the tests similar to those in Table 4.7.

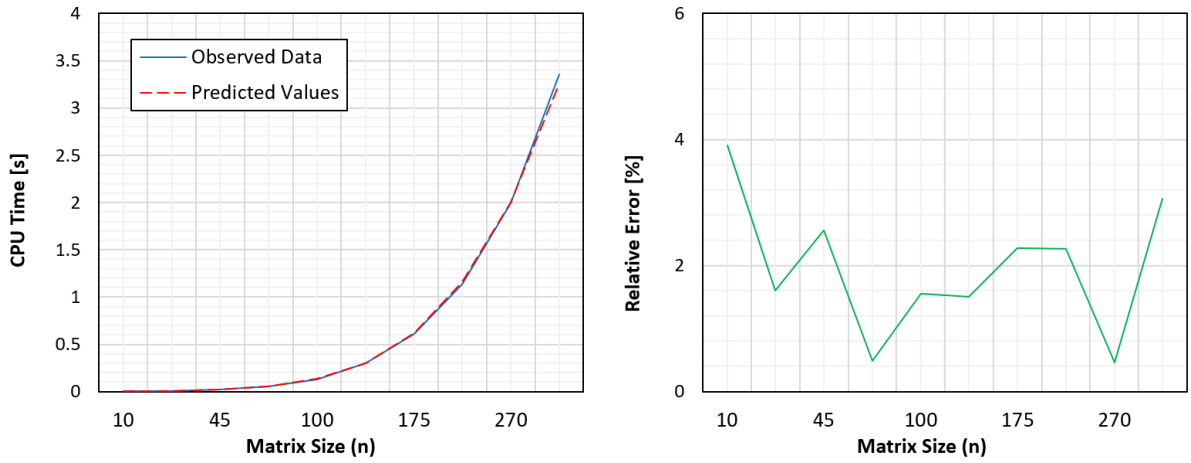


Figure 4.25: Testing Phase in Adaptive Optics

In turn, the proposed models for calculation time with reduction methods can be used to estimate the reduced matrix size needed to maintain the same calculation time while increasing the maximum radial order. As a given example, let the maximum be equal to ten. One can employ *BMR* in order to compensate for the increase in calculation time. This would be based on the calculation time model for the resulting matrix size before and after order increase, then finding the value of k needed to maintain a constant time. Figure 4.26 shows the reduced matrix size needed for each increase in maximum radial order. The reduced size needed rapidly decreases to maintain the same calculation time in turn rapidly increasing the Hankel reduction error.

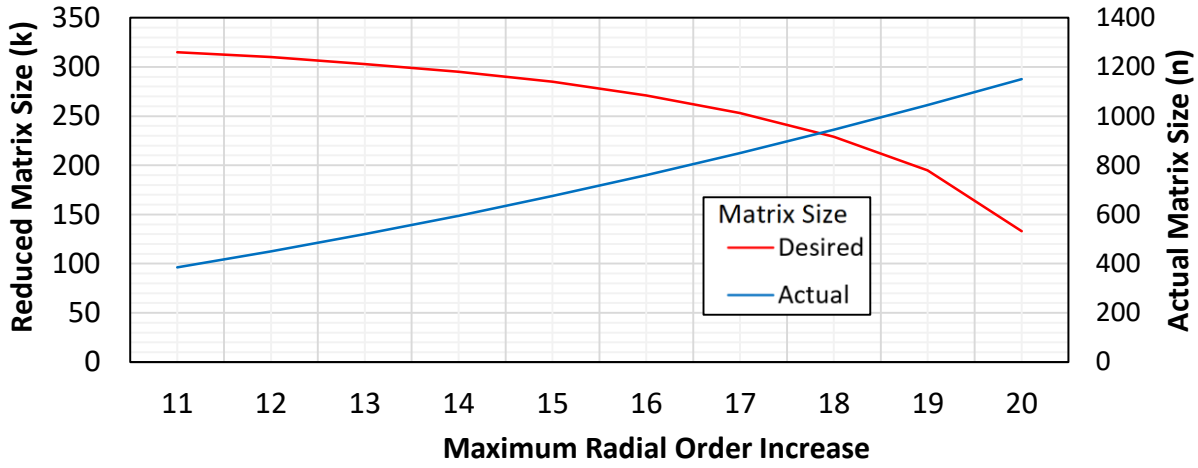


Figure 4.26: Required Reduced Matrix Size

4.7.4.1.2 Cost Approximation for Fixed Parameters

For this simulation, the maximum radial order is set to thirteen, resulting in $n = 520$. The use of the previously defined cost function allows the choice of the target reduced matrix order k as the optimal balance between precision and calculation time. The optimal value is found as $k = 184$. Figure 1.26 depicts the variation of the cost function in function of k. The optimal reduced matrix size is found less than half of the original size which indicates a rapid decay rate in Hankel Singular Values. This can be the result of the large amount of zeroes in (4.64).

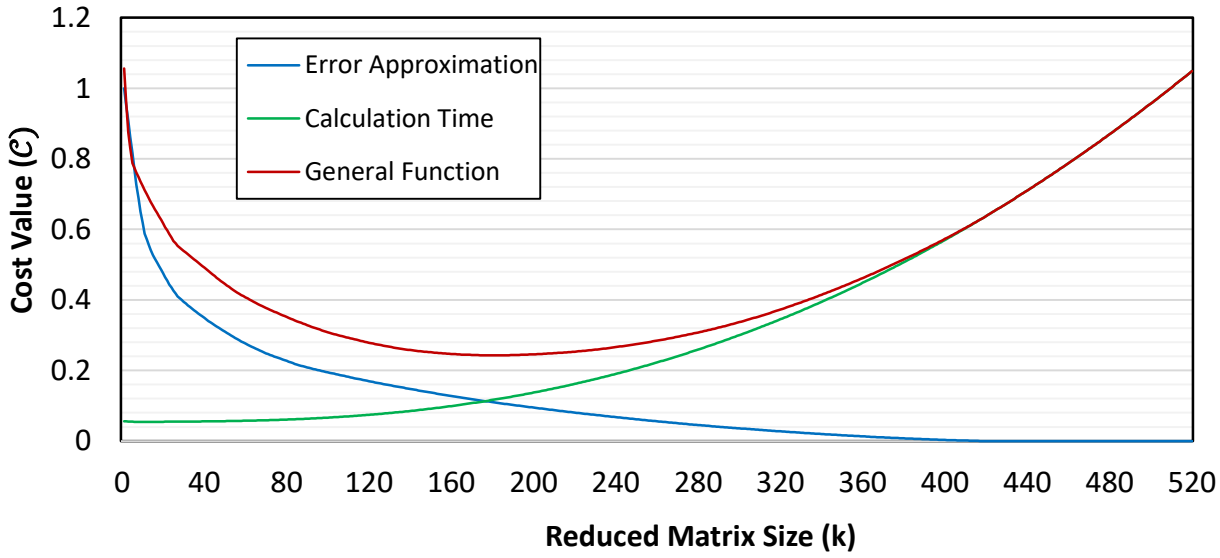


Figure 4.27: Variation of the proposed Cost Function

The provided Cost Function offers the possibility of optimizing the parameters of the system in order to balance between precision and calculation time. In the case of Figure 4.27, the presented results are for fixed AO parameters where the study focused on the reduced matrix size. However, the results presented can be extrapolated to provide an educated study of the AO variables.

4.7.4.1.3 Analysis of Parameter Influence on Cost Approximation

In the case below, the shift focuses towards the physical parameters in the domain. The objective is to study the influence of each parameter on the resulting cost function and in turn the optimal reduced matrix size. This variation is primarily represented in A_r in (4.64).

The first study is based on the variation of the maximum radial order from thirteen to 16, therefore increasing the matrix size from $n = 520$ to $n = 760$. Figure 4.28 depicts the difference in the resulting Cost Function between the first and the new matrix size.

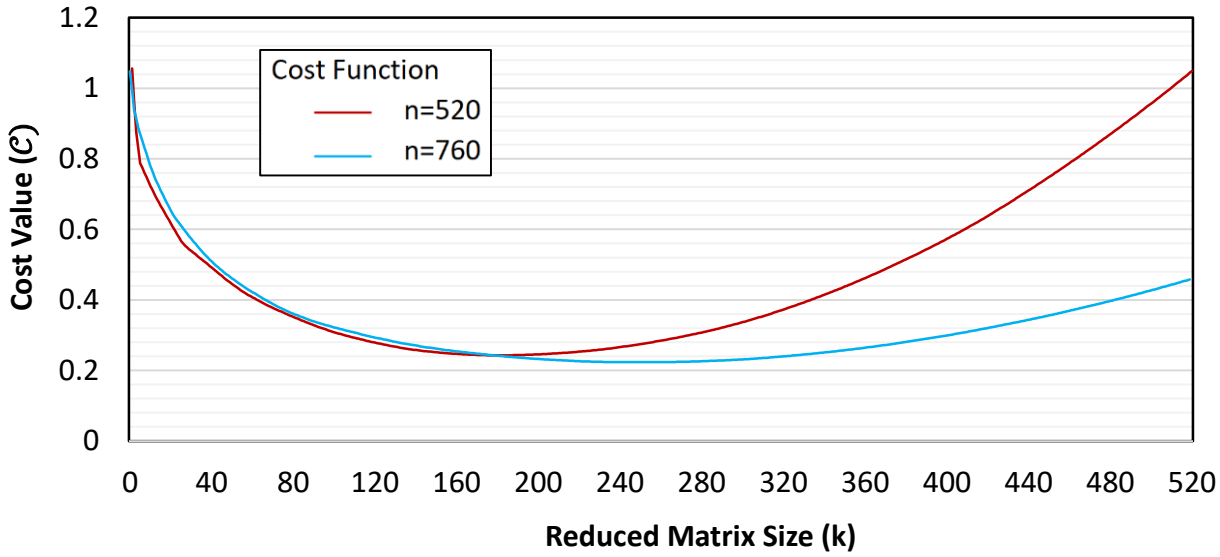


Figure 4.28: Comparison between the different Cost Functions

The results show that for the previously stated system properties, an increase of 46% in matrix size has led to an increase of 39% in the optimal reduced matrix size. This difference opens the potential to balance between the increase in precision from the radial order and the decrease from matrix reduction where the decision of changing the AO parameters are based on the user/researcher's preference for the output of the system in control. The same process can be applied for other parameters such as number of layers, guiding stars and other and offers the option to balance between the variation of these parameters and resulting cost function depending on the preferred properties of the AO control system.

4.8 Conclusion

The presented models in this chapter allow an educated prediction of the time complexity of the Riccati Process employed in LQR/LQG control alongside establishing the relationship between the gain in calculation time and loss in precision when applying matrix reduction methods for multi-dimensional state-space dynamic systems. The primary note is that these models also offer an observation of the variation in time complexity via a polynomial function given a set point of precision on a given processor. This is apparent in the case-study taken where the cost for increasing precision through the addition of sensor points or additional setup of layers can be analyzed.

5 Prototype Application

5.1 Introduction

The developed prototype application for testing is a smartphone application for the domain of Control. Its core is in 2 parts; the first is the Model Identification process and the second is the control process. The development and design of the application is based on the solutions provided in the previous chapter in order to bypass the challenges that are experienced by other scientific smartphone applications. Its aim is to offer the user the ability to identify a model for a studied system based on the collected Input/Output Data and to insert this system model in a control scheme for simulation. Figure 5.1 shows the software map for the User Experience.

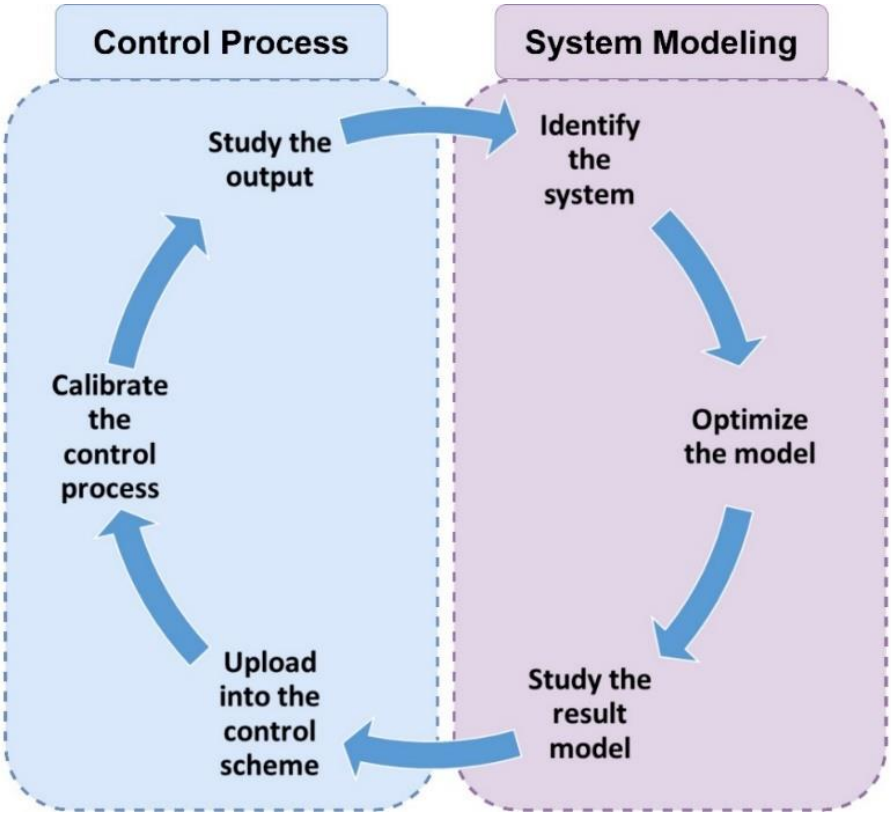


Figure 5.1: Application Software Map

The subsections of the application are also split based on the shown software map where the aim is to tie these subsections to smoothly transition the user from the start to the end. Figure 5.2 shows the user's roadmap through the different main sections of the application.

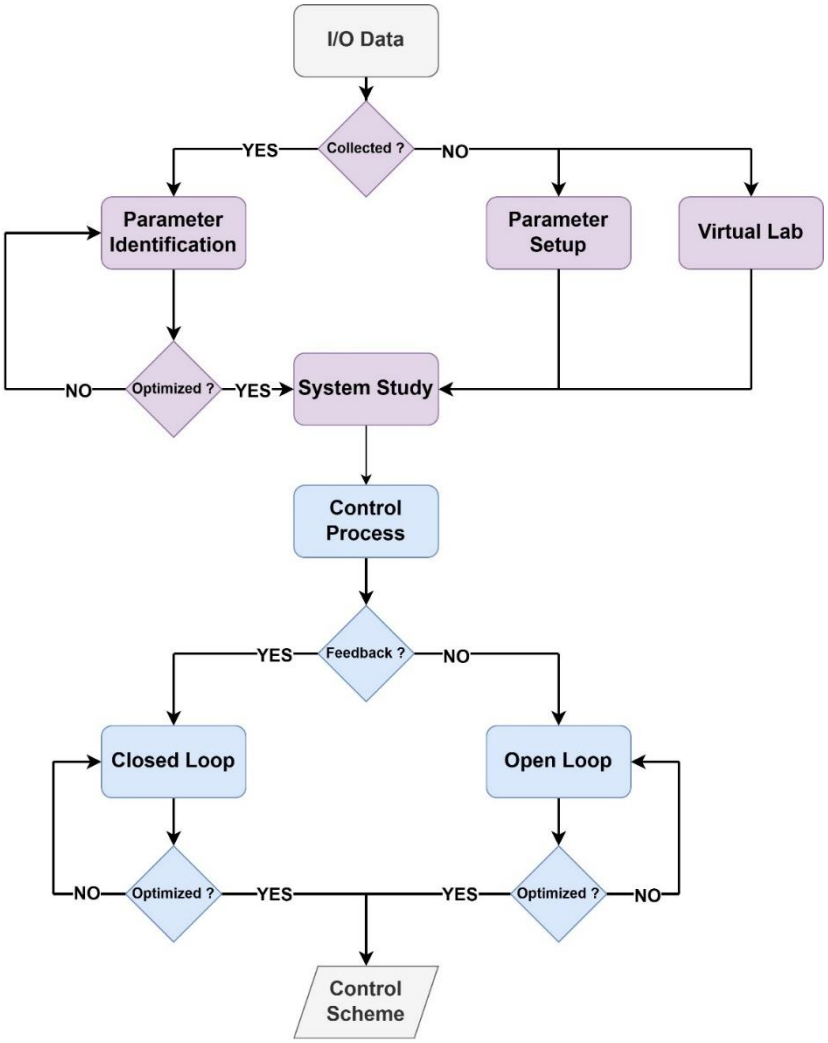


Figure 5.2: Application subsection task division

The application allows the user to either import Input data and Output data from a recorded experiment or simulation or to generate this data based on the chosen model, parameters and other factors specific to the model. The user can also access built-in examples in the Virtual Lab section to be used later on. Afterwards, several sets of standard studies and tests are provided in order for further testing of the model such as Nyquist test, Lyapunov stability test or other. Then the studied model is transitioned into the control process where a set of control schemes is offered to the user. After choosing the control schemes, the user chooses the specifics of the control scheme such as Controller type and reference type alongside calibrating the parameters of the blocks.

5.1.1 Purpose of Control Application

The advantages of employing such type of a simulation application on an accessible platform is mainly from two perspectives; educational and engineering.

Smartphone dependency in education has increased where the concept of ubiquitous computing is being slowly implemented in higher education through smartphones [153]. The social aim of the application is to offer a portable simulation environment for students in order to easily

test learned concepts and presented experience without the need for stronger hardware/software. In addition, the application also contains a set of built-in examples that cover several aspects and scenarios which in turn acts as a virtual laboratory for control simulation.

From the perspective of the industry, mobile applications have shown development alongside a promising future towards varying engineering fields. However the main increase in engineering dependency has been apparent in field service engineering domains due to the smartphone's ability to alleviate the limited on-site information availability. Several applications that target a specific domain of field-service engineering have emerged and maximize on the options provided by the technology implemented in smartphones. These applications focus on investigating problems and troubleshooting alongside documentation and service report. Since engineering applications are developed with focus on one specific domain, prototypes are constantly being developed and marketed in order to test the reaction of the concerned on-field engineers [154]. A control application allows the user to directly import recorded Input/Output data from an observed system, conduct black-box identification of the system and simulate the control process of this system all during field work. In addition, it also allows the user to conduct stability tests for the system for both safety measures and performance optimization.This chapter is split into four sections. Section 5.2 focuses on the design of the UI. Section 5.3 concerns the development and implement of the System Modeling segment of the application and Section 5.4 concerns the control process section. Section 5.5 focuses on the Virtual Lab section.

5.2 User Interface Design

The approach for the GUI design, shown in Figure 5.3, is based on the solution proposed for the challenge of optimization of space memory. In addition, the GUI must be designed in a way that enables the implementation of the other solutions as well. Apart from addressing the challenges, it also must be designed in a way to guarantee optimal user experience and ease of learning.

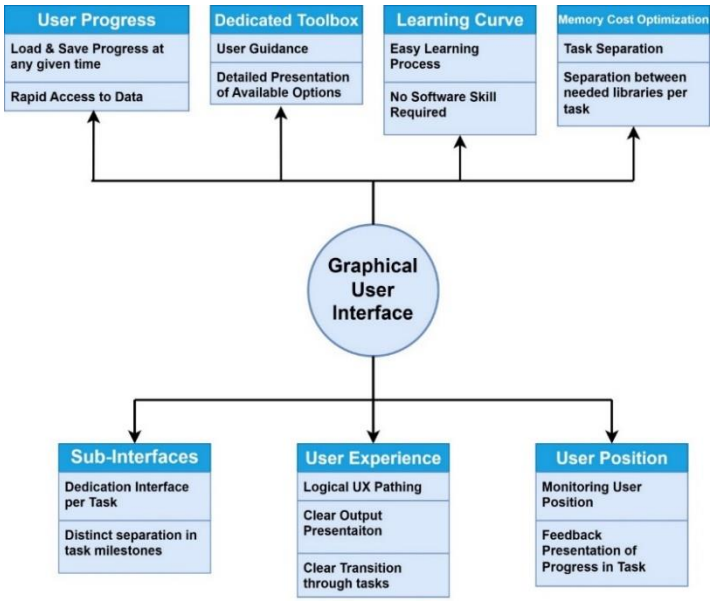


Figure 5.3: User Interface Criteria

Each element of the GUI follows a modular design that allows its adaptation and execution in several scenarios without reducing the efficiency of the execution of the algorithms. The combination of these core basis allows the application to offer a good User Experience while ensuring the respect of the smartphone's hardware/software constraints. In addition, it removes any skill requirement for conducting operations on the application unlike most other desktop applications in the domain. This is also ensured by offering a linear and conventional distribution of tasks that is based on the expected route taken by any user that is involved in the domain. The modular design also allows the execution of each task both separately and as a combination from a larger user project. This permits the application to operate both as a rapid support for smaller tasks as well as the setup of large scale projects and control processes.

In order to ensure ease of learning, the sub-interfaces follow the same template that contains the listed criteria and supports the sequence of operations from the start of the task to the end. Figure 5.4 shows the adopted blueprint for the sub-interface where each following subsections explores its elements.

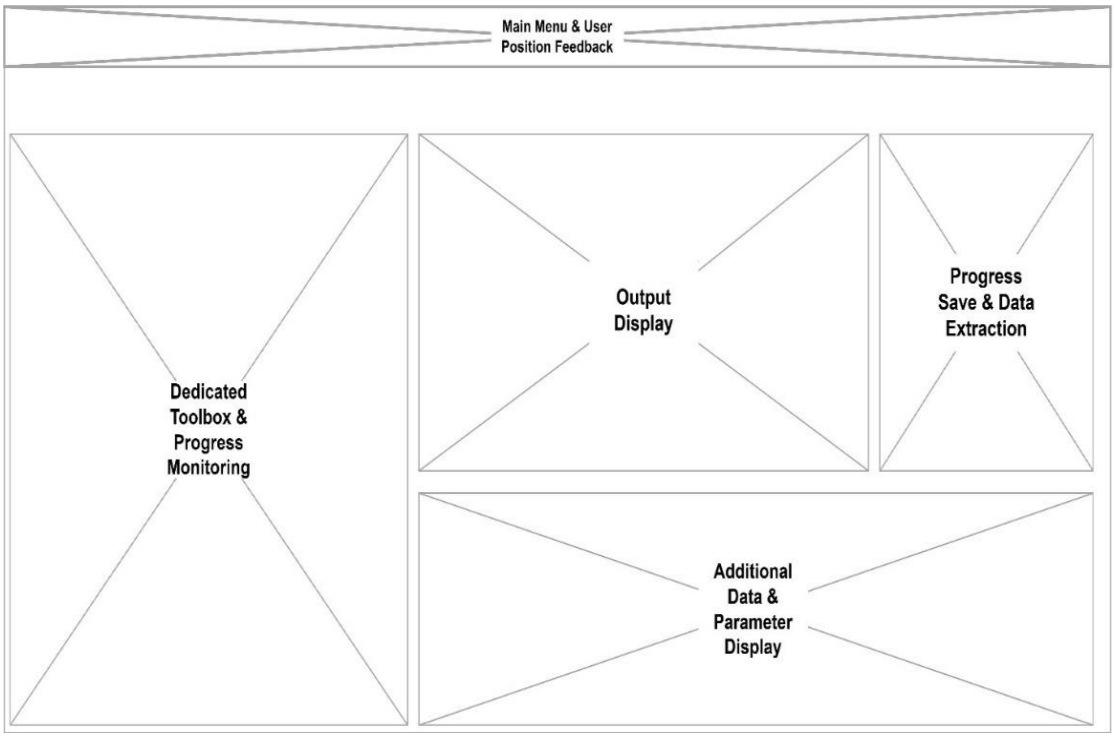


Figure 5.4: Application sub-interface blueprint

5.2.1 Main Menu

The main menu, described in Figure 5.5, acts the primary link between the user and the remaining operations in the application. The main sections of the menu are displayed in the sequence that the user is expected to follow throughout the application. The user starts at the home section then proceeds to System Modelling and afterwards to Control Process. In addition, a system model generated in System Modelling is needed first to be inserted in Control Process.

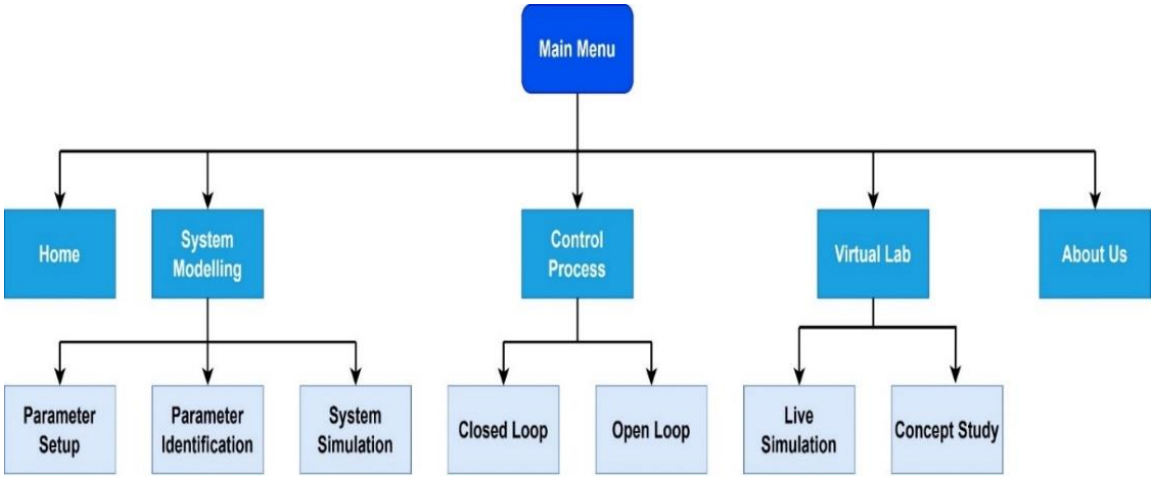


Figure 5.5: Application Main Menu Divisions

5.2.2 Progress Save

The Progress Save section, shown in Figure 5.6, is for the user to save his designed model. This model can be either saved as a Sensor or a System to be used in the Control Process. The Save section is placed right next to the output display in order for the user to intuitively know the next step without any need for extra-guidance or application intervention.

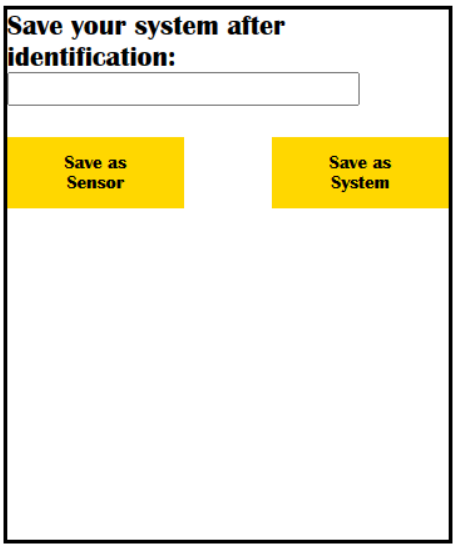


Figure 5.6: Application Progress Save Section

Each model is assigned a specific name and the Model data is extracted directly from the simulation which primarily include Model Parameters such as Output and Input parameters in ARX model.

5.2.3 Output Display

This section is for the display of the output. It is primarily dedicated for graphical display of the results alongside other data such as observed output also in the case of model identification. Figure 5.7 shows an example of the output display.

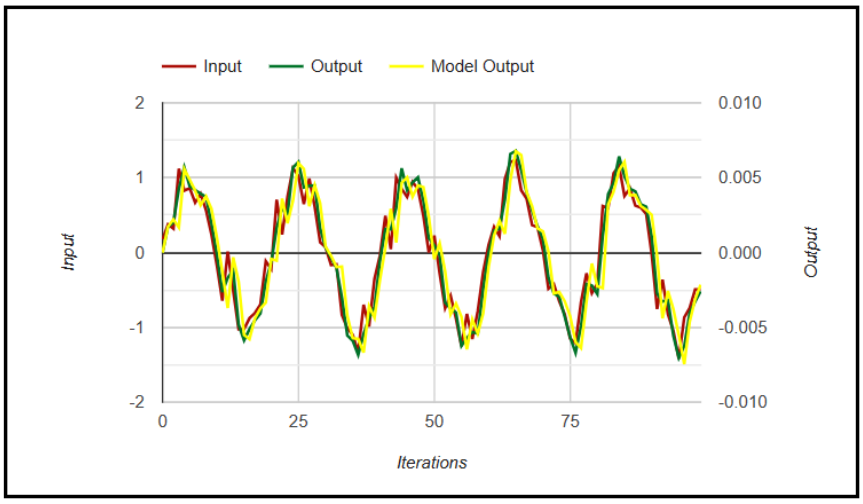


Figure 5.7: Output Display Example

5.2.4 Additional Data Display

This section is used to display any supplemental data to the operation conducted such as the observed mean error and calculated parameters in the case of Model Identification. Figure 5.8 offers an example of this in the case of Model Identification using the previously proposed identification method.

```
Input Parameters: 0.006
Output Parameters: -0.191,-0.166
Mean Error: 0.623
Maximum Error: 7.99
BEST ITERATION: 24
ERROR TICK:
27

0.9653469895787927
0.9653469895787927
1.234198787794214
0.8140654204513689
0.8360991461853488
-1
0.9950820708466525
```

Figure 5.8: Additional Data Display Example

5.2.5 Dedicated Toolbox

The toolbox design follows a sequential filtering of options from the start of user calibration to the end. The toolbox must cover all of the standard implemented options while not overwhelming the user with all of the options simultaneously. The approach would be a systematic approach that would sequentially filter the user's options based on their step-by-step input and choices. An example of this is Model Identification where the user starts first by choosing whether the requested model is Linear or Non-linear, then proceeds to choose the type of the model as in the case of linear ARX, ARMA, ARMAX, etc. and after that he chooses the Model Identification type then the Input/Output parameter size. Figure 5.9 shows the Model Identification Process where the user chooses a Linear Model then an ARX model for model identification.

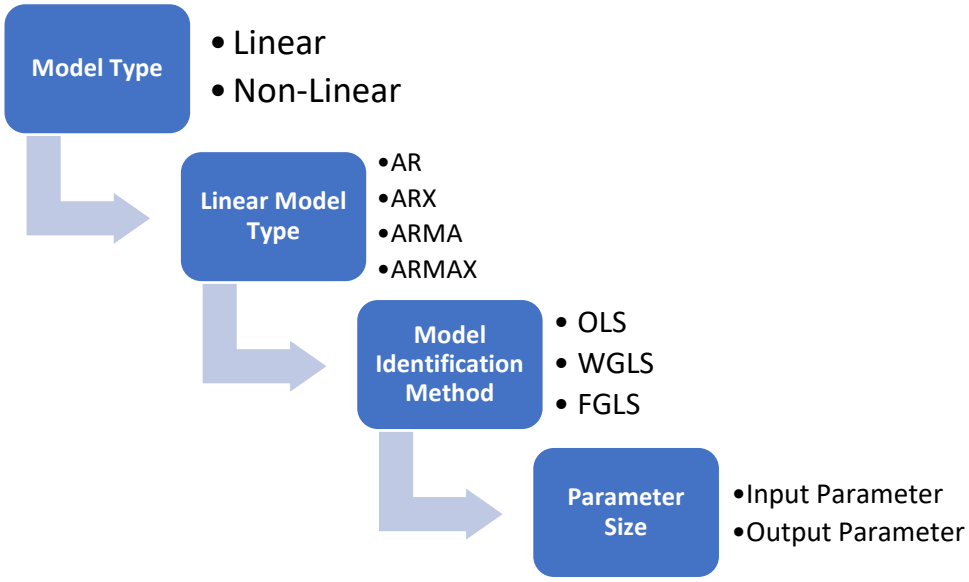


Figure 5.9: Model Identification Toolbox Sequencing Example

This Toolbox template is specifically designed for each section of the application. This process clearly requires the knowledge and study of standard used processes and operations by the developer in order to cover the work that is expected in the domain. This design choice acts as the primary solution for the previously listed challenge of "Removal of Dependency on Programming Languages". This removes the need for the design of a user-oriented programming library while increasing the development cycle in order to add the expected options. Figure 5.10 offers an example of the toolbox for Model Identification of a ARX model.

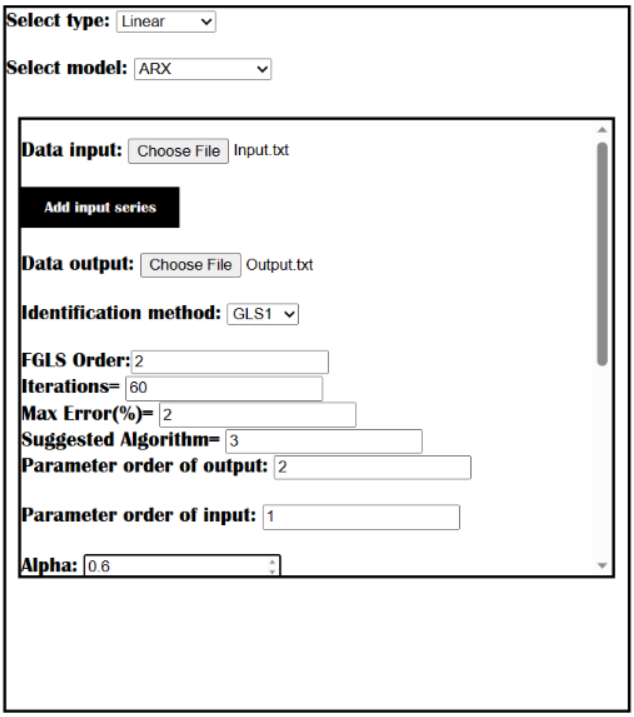


Figure 5.10: Model Identification Toolbox Example

In addition, the toolbox is also used to monitor the user position in the task cycle as well. In an example of Control process, the toolbox also highlights the different blocks of the control scheme to be processed by the user. Figure 5.11 shows the Toolbox map in the control process.

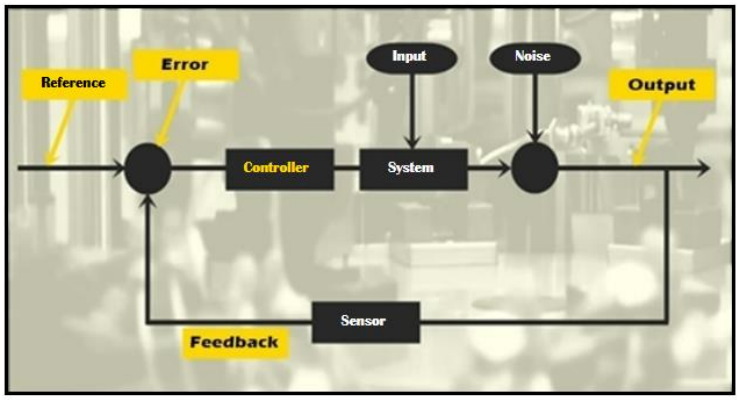


Figure 5.11: Control Process Toolbox Map Example

5.3 System Modelling

The main focus of System Modelling, described in Figure 5.12, is the design of a model that can be used later on in the Control Process where it can be saved as a system or sensor. This section presents the varied options presented in order to successfully conduct this modeling process.

5.3.1 Concept of Process

This model can be either manually created by the user based on parameter setup or identified based on the uploaded Input/Output data. In addition, further simulation can be conducted based on the resulting model such as testing under different input data or Unit Step Response Testing.

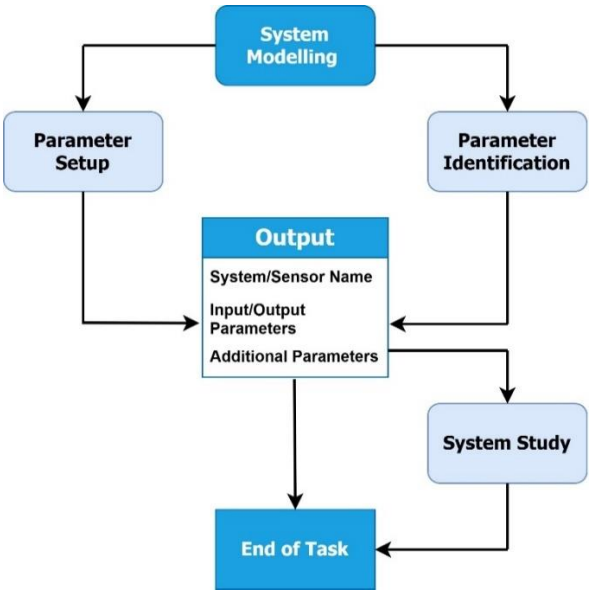


Figure 5.12: System Modelling Map

5.3.2 Identification of System Characteristics

In this section, the user can insert the Input data and Output data of an observed system in order to identify a model. Figure 5.13 presents the options offered to the user. These are first divided into Linear model or non-linear model. In the case of Linear model, the next step for the user is to choose the model type (AR,ARX,etc.) and then moves on to choose the identification method (OLS,FGLS,etc.). Afterwards the user chooses the parameter number for both the Input Parameters, Output Parameters and Noise Parameters (in the case of ARMA, ARMAX). In the case of Non-linear model, the model type is based on the Model Identification method such as piecewise linearization where the resulting model type is a set of linear ARX models.

The displayed output are Predicted Output, model parameters, mean error and maximum error. The mean and maximum error are displayed in order to compare with the uploaded observed output in order to assess the efficiency of the identification process. Based on the output, the user can re-calibrate the parameter size and model type in case the observed error is not sufficient. The predicted output is displayed with the observed output in order to enable a visual comparison of the result.

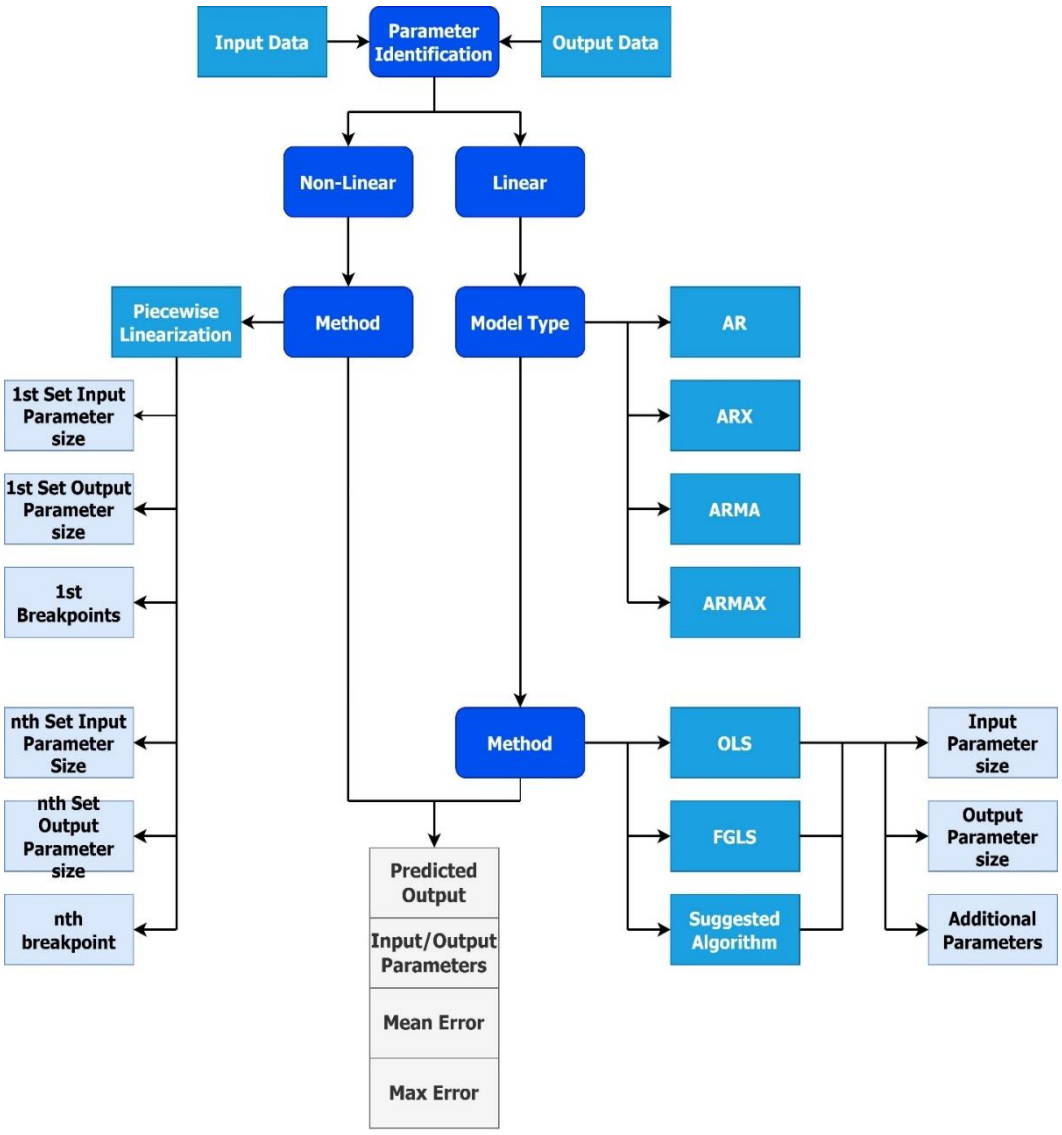


Figure 5.13: Parameter Identification Map

The example below shows the identification of a Centrifugal Pump system. The model to be identified is a linear ARX model and the chosen method is the previously explored suggested algorithm for model identification. Figure 5.14 shows the application output.

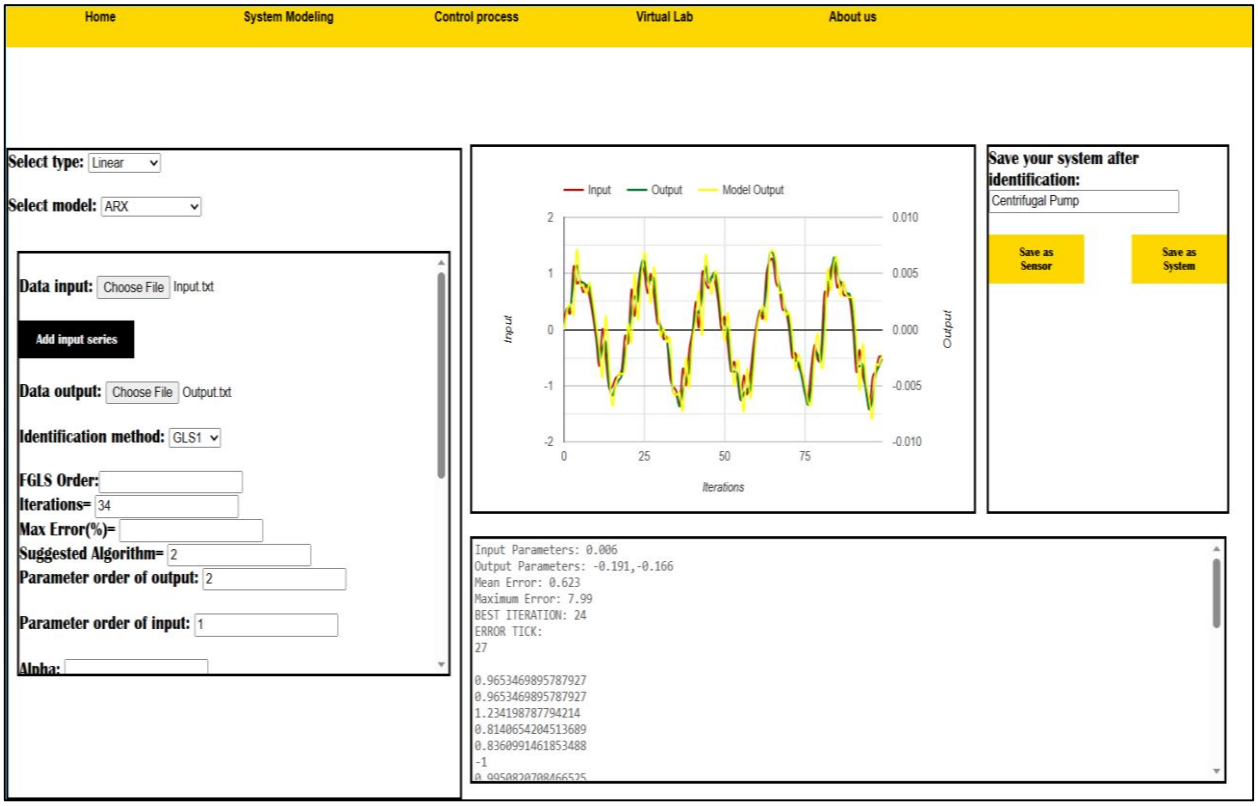


Figure 5.14: Parameter Identification Application Example

5.3.3 Parameter Setup of Model

In this section, the user manually sets up the parameters of the targeted model. This offers the inverse approach of the Parameter Identification process. Figure 5.15 describes the steps conducted in this section. The user begins by choosing the desired model (AR, ARX, etc.) and then moves on to choose the parameter size and in turn choosing the related parameters. In addition, the user also has the option to generate noise (Poisson, Gaussian, etc.) in order to impact the generated output. The model can be saved to be used later on as well whether as a system or a sensor.

This section also allows the user to copy the data that was generated from the Parameter Identification section and calibrate some of these parameters in order to test a specific scenario.

An example of this is identifying a ARX model in the Parameter Identification section and then testing the variation in the system if the output parameters were further prioritized.

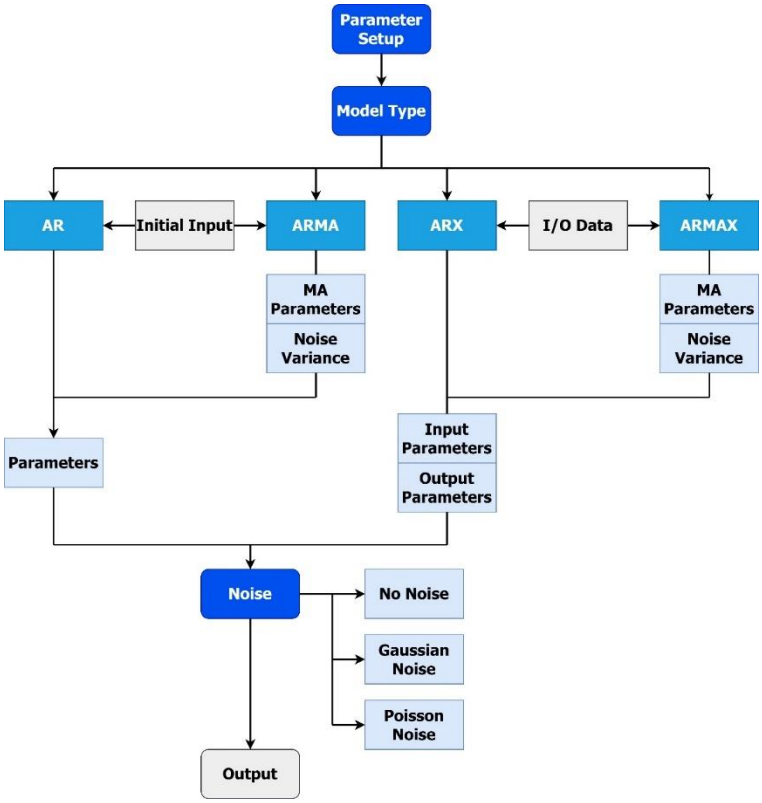


Figure 5.15: Parameter Setup Map

An example, shown in Figure 5.16, is presented below which follows the design of an ARX model. One of the difference between an AR model and an ARX model is that the ARX model requires input data for the length of the iterations. Afterwards the user chooses the order of input parameters and that of the output parameters. Then the user calibrates the input/output parameters and proceeds to add noise.



Figure 5.16: Parameter Setup Example

5.3.4 System Simulation

In this section, the user is able to upload one of the previously designed system/sensor in order to be further tested whether through a standard test (unit step response) or to be tested with another set of input data in the case of ARX/ARMAX as seen in Figure 5.17. This section offers the ability to further study the design to assess whether there is need for further calibration.

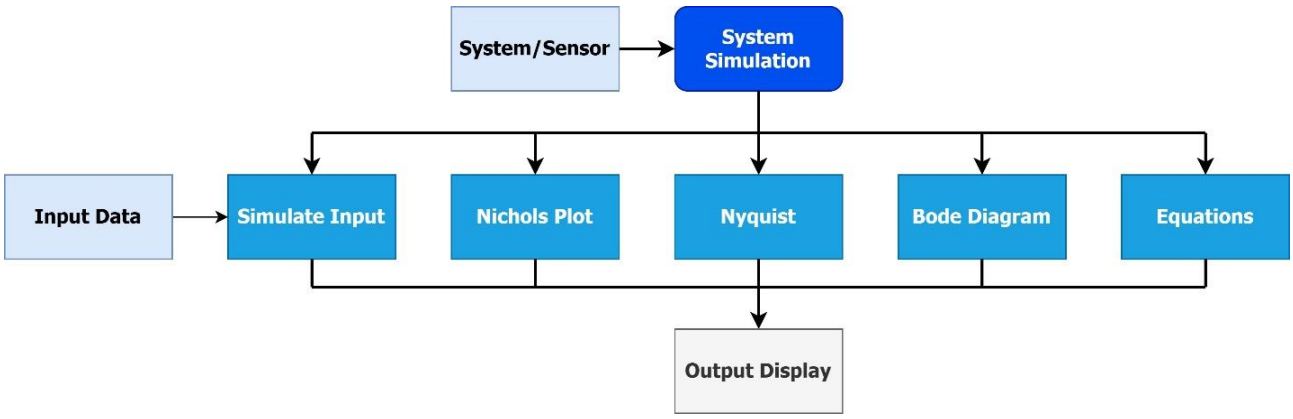


Figure 5.17: System Simulation Map

The following example, seen in Figure 5.18, shows the upload of a previously designed system to be tested with a new set of input data.

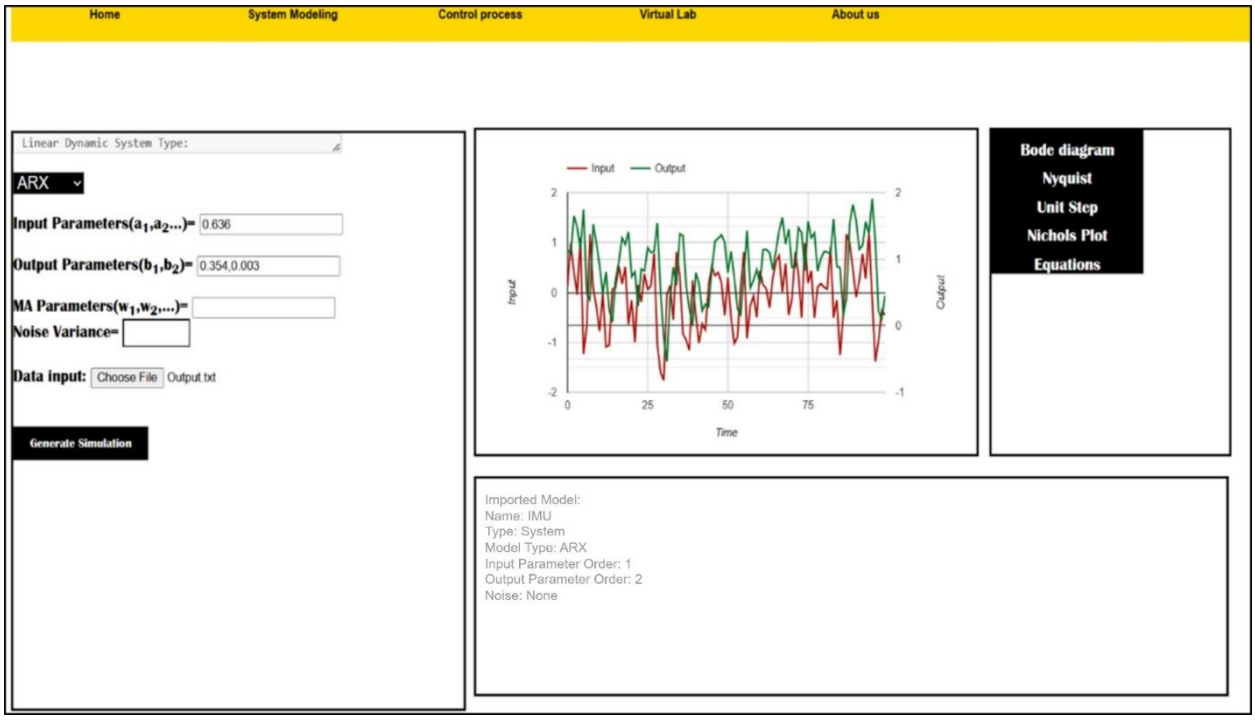


Figure 5.18: System Simulation Example

5.4 Control Process

After the user finishes the design of the system/sensor and it is saves, the user can then move on to design the control scheme. This design includes the setup of the reference, the sensor and the controller. Several control schemes have been developed and utilized throughout the domain of control and vary from standard conventional design to control schemes designed specifically for a given system.

As explained before in the Challenge of "Removal of Dependency on Programming Languages", the development phase first focuses on the development and implementation of the mostly used and common control schemes and branches out to more specific schemes. Figure 5.19 presents the options provided in the Control Map. All control schemes have the main objective and direction of operations and are divided into 2 main sections; closed loop control and open loop control. In closed loop control, the output directly impacts the input of the system unlike in the case of open loop control. The parameters that need to be calibrated by the user vary between each set control scheme whether it is because of the number of elements, their position or the setup of the blocks. In addition, the parameters vary based on the type of specific block chosen such as the parameters specific to PID controller and those specific to LQR controller. The Control Process is divided into two main sections which are Closed Loop Control and Open Loop Control.

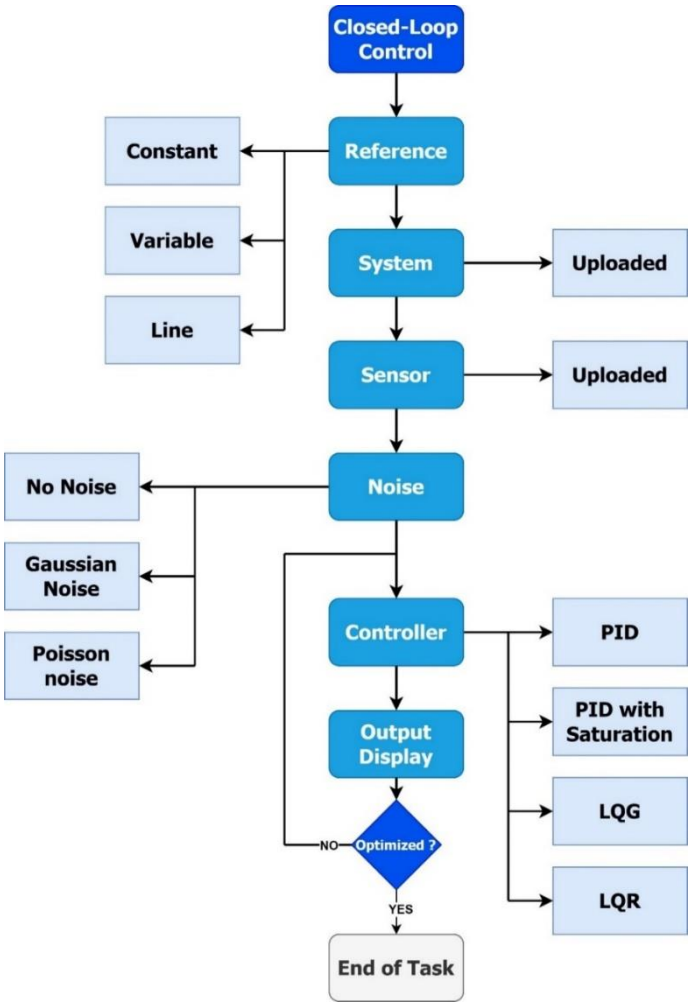


Figure 5.19: Closed Loop Control Map

In the Closed Loop Section, the user first starts by choosing the control scheme that suits his desired project. The center of the scheme is the system that was previously designed by the user in the System Modelling section.

After choosing the control scheme, the user proceeds to choosing the controller. The choice of controller influences the future parameters to be calibrated. The user then uploads the system and the sensor and moves on to the design of the reference. Noise can be added to the output of the system.

In the example shown in Figure 5.20, the user designs the control process for the "Lowpass Filter" system. The user starts by choosing the type of Reference which is a constant reference. Afterwards, the user chooses the controller type which is PID and proceeds to calibrate the controller parameters. Then the system is uploaded and a sensor as well which is a delay sensor.

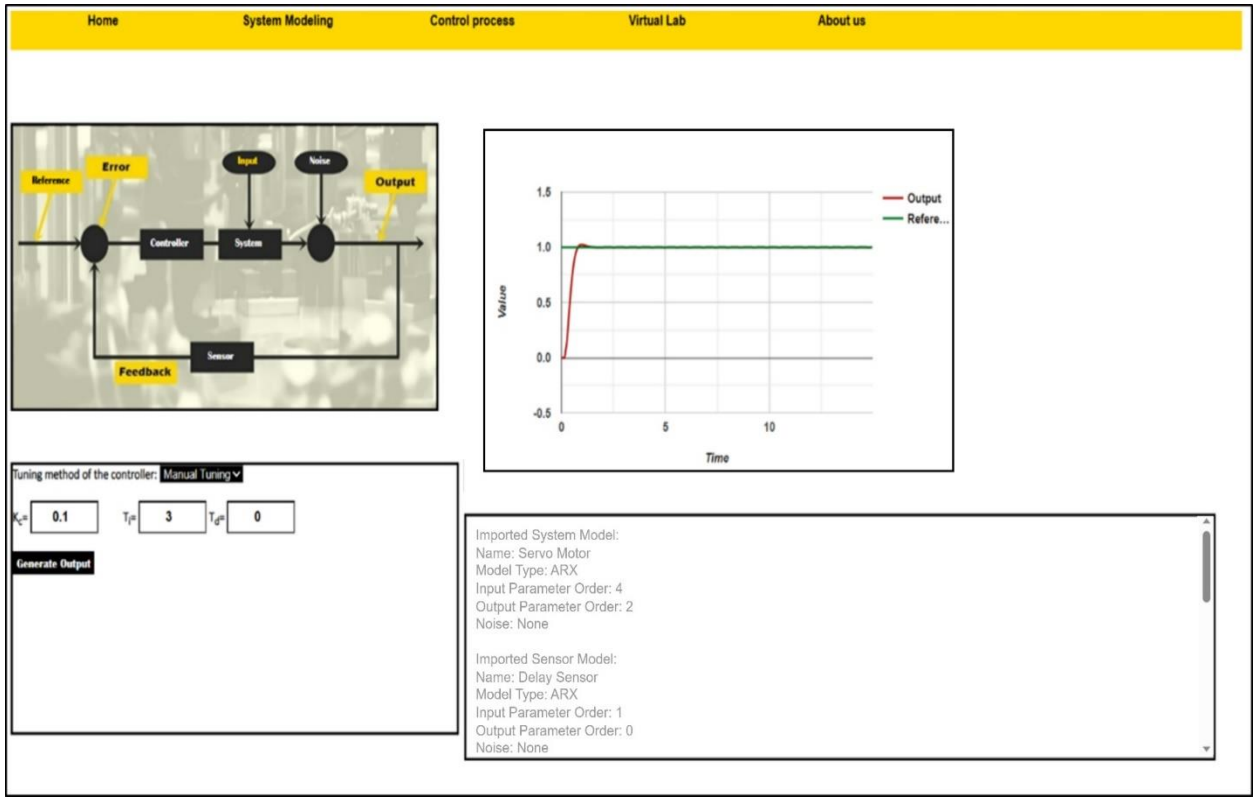


Figure 5.20: Closed Loop Control Example

The second part of the Control Process is Open Loop Control which is described in Figure 5.21 and differs from the previous section by the fact that the control process does not include feedback. The controller input is purely based on the reference.

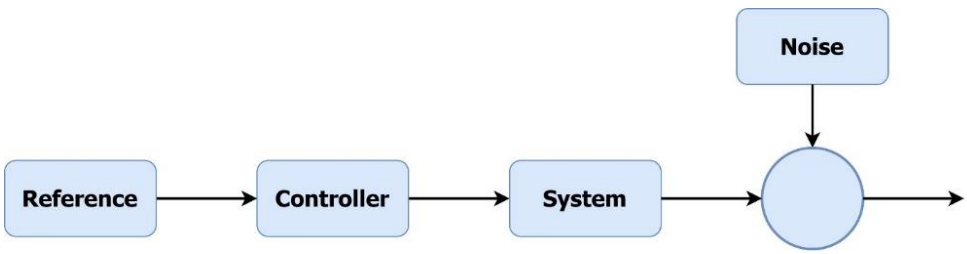


Figure 5.21: Open Loop Control Scheme

5.5 Virtual Lab for Smartphone Simulation

This section aims to show the development of the concept of virtual labs in the field of engineering on the smartphone platform. The concept of laboratories is one of the pillars of scientific work, whether at the educational level or in the field of research. The original reasons for focusing and developing laboratory work lie in the need to produce highly skilled technicians for industry and highly qualified workers for research laboratories [155]. However, the increasing demand for conducting experiments at the university shows the importance of laboratory work not only for research but also for education. The need for divergence and expansion of this field aims not just to enhance student performance, but also their perception. The risk in laboratory work is that it can provide a false perspective of the goal which leads students to believe that laboratories only aim to prove previously taught theoretic concepts [156].

The idea of Virtual Lab developed as one of the main visions of lab work as a whole as it aims to provide the user with a proper and detailed lab experience through a broader alongside providing a proper presentation of all of the required aspects for simulation and experimentation. While laboratory work is common for both educational fields and the research industry, virtual lab targets students through a set of key examples and demos that encompasses the different variables that define the main principles of control study.

This section aims to show the development of the concept of virtual lab on the smartphone platform. This starts by exploring the state-of-art of the concerned subject and moves on to highlight the advantages, disadvantages and the challenges. Afterwards, the implementation of Virtual Lab in the prototype application is presented.

5.5.1 Overview on Existing Lab

The following overview is conducted to explore the progress in the field of lab work which highlights the development in used technology.

5.5.1.1 Live Physical Lab

While lab work has been a staple of science since the classical ages such as with alchemy, it was on the basis of empirical research that originated more relevant than theoretic work in the domains of natural science and as an individual motive not as an institutional entity. The concept of laboratory as known today emerged to relevance in the last third of the 19th century where buildings specially designed for this purpose evolved into central institutions of scientific endeavors where the involvement no longer strived for formation of individual character as was the case before but where the work was executed by professionals that applied standard methods with the aim of achieving innovation [157]. The second transition of laboratories into becoming a staple of academic studies in science began in the second half of the 20th century where there was a growing set of studies that claimed that the traditional sense of science instructions was showing inefficiency in changing students' misconceptions of the world around them alongside the failure of transitioning scientific learning from the classroom context to the physical world [158]. Therefore it can be seen that the impact of laboratories does not only stem from the idea of future potential research career but as a basis for all fields

of natural sciences. In the particular case of engineering lab work, the core function of the engineering industry is the manipulation of energy, materials and information and in order to successfully achieve this, engineers must have a knowledge of nature that exceeds the concepts of theory and that is gained through educational laboratories [159]. The type of work greatly varies between the engineering disciplines where some lab work is based on the use and consumption of sources and materials such as in chemical laboratories or in the use of powered equipment with less focus on material usage like in electrical laboratories.

5.5.1.2 Lab at Home

The main concept of lab at home is based on the idea of providing students with the required material and equipment to conduct safe and minor laboratory tasks outside the perimeters of the university laboratory. While this idea has been present for a while, it became relevant with the lockdown crises that forced universities to change their approach to operations. The COVID-19 pandemic has forced educators around the world to adjust their teaching techniques, to accommodate the strict social distancing measures required for in-person interaction, as well as completely redesigning some activities for remote delivery [160].

The purpose of such type of work is the same of that of the physical laboratories as stated above alongside promoting student independency and self-reliability. In addition, students are introduced to accessible equipment that allows them to improve their discipline even outside of the obligations and schedules. For example, the Arduino Kit in Figure 5.22 [160] provides students with the ability to create and develop new tools and products with a relatively cheap need for equipment and resources.



Figure 5.22: Arduino Lab Toolkit

Studies have shown that students respond equally to remote labs and lab work in the institutions including several shown advantages which include safety, scalability, self-enhancement and the potential of cooperation between institutions [161].

However, this type of emerged when the development of virtual education and simulation drastically increased as well which lead students to seeing the use of lab kits as an extension of the methods of education and vice versa. Therefore the lines and differentiations between the two methods are, simply put, vague from the perspective of the students. It is to be noted as well that this type of laboratory work suffers from one of the main concerns of the previous version which are the high expenses even if they tend to be directly divided on the students and

not the sole responsibility of the host institution. In addition, because of the lack of on-site instructors for user, lab scalability and variety is limited in this type in order to remove any potential safety hazard risks [162].

5.5.1.3 Remote Lab

The previously stated lab types primarily target academic institutions and enhance student performance and engagement. The concept of remote control lab is witnessed in professional research fields especially in the cases of operating dangerous equipment and materials. This type of work was developed with the aim of obtaining a physical output. An example is seen in Figure 5.23.



Figure 5.23: Remotely controlled lab kit

For such reasons, this type of lab work does not present any benefit over virtual lab from the perspective of the students. It is also more expensive because of the need of physical materials and resources. Development in this type of laboratory work is aimed at industries in which emphasis is placed on the output of laboratory work rather than individual development and user enhancement.

5.5.1.4 Virtual Laboratories

The concept of virtual lab work arises seeing the technological advancements and to counter the drawbacks of physical work which are primarily expenses, demands as well as the potential risks and dangers from certain hazards. The resources required for the setup and development of scientific laboratories exceed the scope and capabilities of several institutions especially concerning relatively advanced laboratories.

The solution to this would be found in the implementation of virtual laboratories which can be considered as a vital educational tool that counters the lack of practical experience in education [163]. The costs behind the implementation of virtual lab are evidently non-negligible because of the costs of purchasing licenses, services and products as well as the required hardware and software to support the applications and simulations but minimal in comparison to the costs of the other discussed lab works. In addition, the design and customization potential offered by virtual lab platforms offer the user the ability to experience a large set of varied environments and models. The simulation and analysis capabilities are presented with the purpose of experiencing complex systems as well as testing new methods of operations or resources where

the primary benefit would be gathering a broad knowledge and set of data without the need to modify an actual physical system which would lead to wasting resources, cost and time [164].

Several applications and platforms have been developed and offer the user the ability to conduct virtual laboratory experiments in varied fields and disciplines of natural science such as virtual chemical experiments, electronics simulations and physics experiments. While the costs are minimal in comparison to other forms of lab types as stated above, virtual lab work relies on dense simulation and compiling which in turn translates into the need for an able hardware/software basis to support these simulations. However, the setup of these types of lab work on personal platforms allowed the user to conduct lab work and experiments from any desired location and at any time.

Therefore the concept of virtual lab work aims to combine the advantages offered by the other lab type without their most noticeable drawbacks and expenses. Some of the relevant applications that support virtual lab applications include Matlab Simulink as seen in Figure 5.24 [165] which offers a dense set of design options and examples that have been constantly developed throughout the years since its creation, P-Spice that specialize in electronic design and simulation and other.

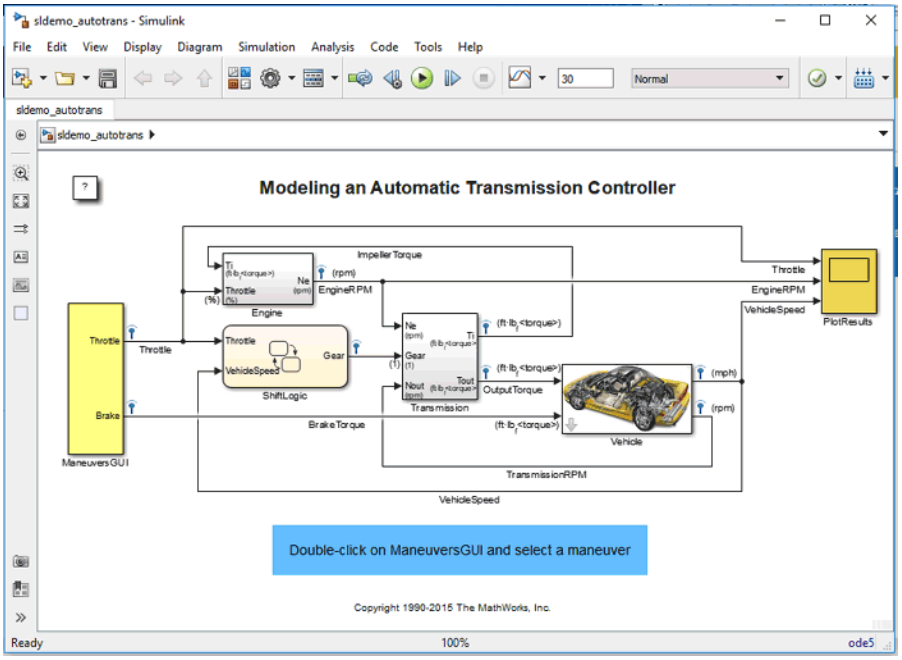


Figure 5.24: Vehicle transmission built-in example in Matlab Simulink [165]

5.5.2 Advantages and Disadvantages

The set of advantages and disadvantages has been classified based on published work concerning virtual labs both for general purpose and for specific educational domains as well. The work of Pearson et al. [166] and several others present surveys that collect a list of advantages and disadvantages surrounding the implementation and use of virtual laboratories.

5.5.2.1 Advantages

Virtual Lab provides the user with access from any practical place that can accommodate the required platform (desktop, laptop, smartphone). It is also not limited to the perimeters of a

physical laboratory [167]. In addition, the smartphone platform both increases accessibility because of its mobile nature and removes any physical constraint.In addition, it facilitates the scaling of the operations conductions where, unlike physical laboratories, an increase in the number of simultaneous users does not dictate the increase of the size and resources provided. Therefore the virtual lab offers a much easier way to increase the volume and number of concurrent experiments [168]. In turn, Eliminating physical resource costs for scalability and operability makes virtual implementation much cheaper than its physical counterpart [169]. Moreover, Virtual labs allow the transfer of saved progress and results digitally and easily among the concerned parties [170].

Concerning the need for supervision, the work of Al Mousawi et al. [171] shows that one of the main benefits of virtual lab work for the students was the removal of the need for instructors and the provided autonomy for design. In addition, Rose [169] shows the ability of virtual labs to track minor data where the user is allowed to focus on the goal of the task. This potential increase in autonomy leads to the enhancement of student's knowledge and understanding of the lab in concern , as seen in the survey conducted by Celine et al. [172].

5.5.2.2 Disadvantages

Most standard virtual lab applications tend to operate based on a specific programming language that the user should be familiar with [173]. In addition, the size and complexity of the application can affect the minimum system requirements to run the application [174].

Moreover, while removing the hassle of manual labor is stated as an advantage, it is nonetheless a nuisance that the student must be familiar with if he wishes to work in a professional research laboratory later on [175].

5.5.3 Challenges for Simulation Design

The following section lists the main challenges concerning virtual laboratories primarily from the development point of view. Further pedagogical research is needed in the field of engineering [176] in order for a conventional set of challenges to be established.

5.5.3.1 Modelling

The modelling approach for an experiment can vary based on the information available for this experiment. There are mainly 2 approaches for modelling which are modelling based on mathematical model and based on collected database. The first modelling approach is taken when the experiment in question is considered easy to implement and execute by the application without requiring high execution costs. This also relies on the availability of a simplified model to be implemented. An example of this is the the work of Arun et al. [177] that focused on the development of a model for the inverse-fluidized bed experiment. When a mathematical model is unavailable or heavily demanding in costs of implementation, an experiment can be simulated based on an associated database that covers the standard output of the experiment in a set domain.

5.5.3.2 Platform Development

The devlopment phase of a virtual lab application requires the cooperation of 2 types of expertise. The first type is in the scientific fields involved, while the second is in computer software and application design. Apart from just a complete understanding of the experiments and theoretic concepts, the scientific team in question must assess the targeted learners and analyze their goals. The goal will be the correct organization of concepts and sub-concepts of laboratory activities [178]. This type of development is expected in any kind of lab work, physical or virtual. The main part of development that distinguishes a virtual lab from other types of lab work is the application developing phases. This includes the development of the appropriate infrastructure that enables the execution of the targeted experiments and simulations.

5.5.3.3 Augmented Reality for Presentation

Extensions and addons should be used to properly immerse the user in the lab work and deliver an experience that replicates the experience of operating in physical laboratories. Studies have shown that animations and visual enhancements benefit student understanding [179]. For such reasons, the virtual lab must aim to provide all of the possible minor elements that aid in delivering a full experience to the user. This can be achieved by implementing visual extensions, notes and animations as seen in figure 5.25 [180].

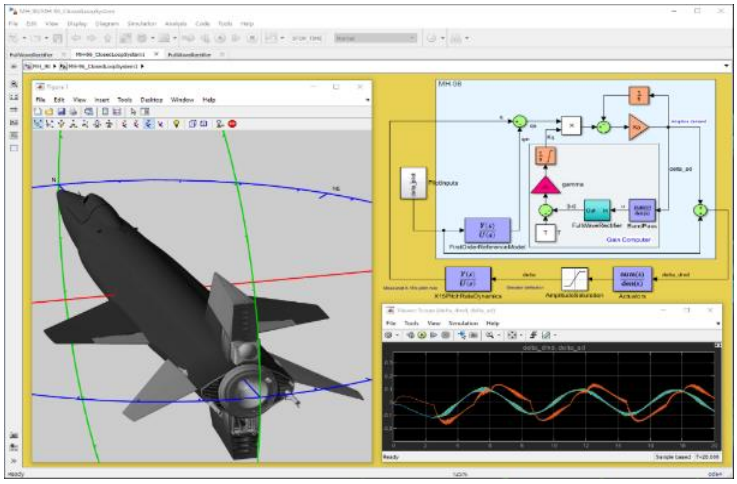


Figure 5.25: Flight control graphical presentation Simulink [180]

These addons show the impact of the user's work in forms presenting the data better than just numerical results. The challenges in presentation are not limited to the platform itself. The presentation provided by the teachers and instructors is also important [181].

5.5.3.4 User Experience

There are 2 knowledge barriers for operating a virtual lab. The first barrier is the knowledge of the experiment and the material in study. The second is the knowledge of operating the virtual lab application [182]. Increasing the customization and the scalability features of the application leads to an increase in the complexity of operations. In order to optimize user experience, UI studies and developments are generally designed with user preferences and tendencies as the highest priority [83].

5.5.3.5 System Requirements

With any application, the hosting platform must satisfy the minimum system requirements. As software projects are developed and increased in feature sizes and complexity, the prediction and control of the resulting requirements must be properly conducted [183]. A potential sudden increase in hardware/software requirements for a virtual lab application can result in lower-end platforms being unable to support the application. Educational applications must be accessible to a wide range of relevant users due to the limited affordability and dedication of a standard student to a single lab course [184].

In the case of virtual lab on smartphone, this challenge lies in developing an application that can operate within the boundaries of the smartphone system considering both hardware and software limitations. An algorithm's performance varies based on external and internal factors. External factors usually vary between platforms and software whereas internal factors are related to the efficiency of algorithms both in memory space usage and time [185]. The potential cost of complex algorithms has led to research that focuses on mitigating the impact. An example of this is Shared-Memory Parallelism [186]. The focus on optimizing memory costs aids in increasing the accessibility of users to the applications on weaker platforms where the operations are considered conducted in a restraint environment.

5.5.4 Elevator Control Simulation

In this section, a set of solutions is proposed for the previously listed challenges and are presented through a developed Virtual Lab simulation in the prototype application. This experiment focuses on simulating the elevator control by introducing varying floor positions and varying loads. The elevator control model is a common model in the field of control. It is characterized by 2 main factors; varying reference and load. The variable reference depends on changing floor requests while the load depends on the passengers. In addition, the organization of the reference is a control study in its own.

In general, elevators follow a logical sorting approach to prioritize which floor requests will be executed first. The simulation offers a detailed experiment that revolves around one of the most commonly used and deployed systems. In addition, it is based on the flow of operations between 2 types of control.

5.5.4.1 Approach for Design Challenges

This section provides a detailed view of the challenges related to the design of this simulation. They are primarily the translation of the generalized challenges in Chapter II into a specific domain alongside the obstacles related to the modelling of physical phenomena.

5.5.4.1.1 Modelling

The implementation of the elevator model relies on the combination of 2 separate work in order to provide for the user the full simulation. The first work focuses on the logical control of the elevator and the search for the Floor Call. The second work focuses on the control of the machine to reach the designated floor that was declared in the Logical Control.

While this experiment contains specific parameters that the user might not be aware of such as pulley width, the parameter data employed and their effect on the experiment are based on the parameters studied in the work used.

5.5.4.1.1.1 Digital Logic

The logic system is based on the work of Adnan et al. [187] as seen in Figure 5.26.

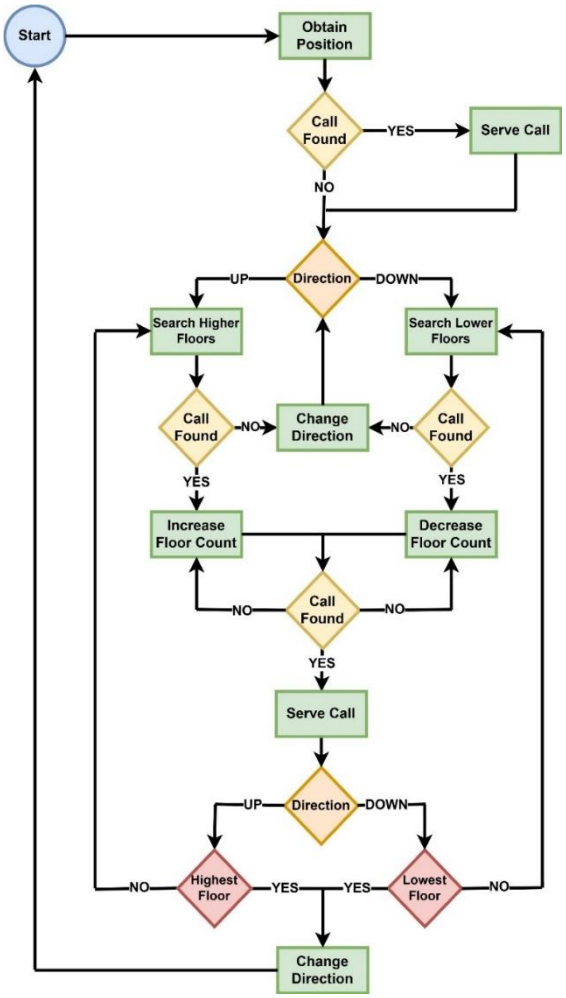


Figure 5.26: Elevator Reference Control Flow Chart

The first half of the control process concerns the sorting of the floor position inputs. This sorting is the reference of the control system. These inputs should be sorted due to the fact that a standard elevator system is susceptible to receive several requests simultaneously with varying costs and time for execution. This sorting enables the execution of these tasks in an efficient manner. A logic system model is implemented in order to conduct this sorting. The idea is to do the cheapest tasks first, then move on to the next. The adopted algorithm starts by searching for the nearest call from the current floor position both upward and downward. If a call is found, it is served and then flushed out of the memory. If not, it moves on to search in the second nearest floor. After finding and serving a call, the same search process is repeated until the upper and lower floor limits are reached. Afterwards, the loop is repeated from the current floor position. A call is served by transferring it as a reference point to the elevator control model. It is to be noted that this logic model shows how the elevator model tends to

drift in one direction to fully execute the calls first. As it executes the first nearest call, the other calls in the same direction become nearer than the calls in the other direction.

5.5.4.1.1.2 Control Scheme

Figure 5.27 shows the cascading control model that simulates elevator drive control. The input of the DC machine is the voltage given to the system. The cascaded controllers act as the transformers of the data given [188]. This model is not the only standard model used, however it serves as an efficient simulation representation of the system in study.

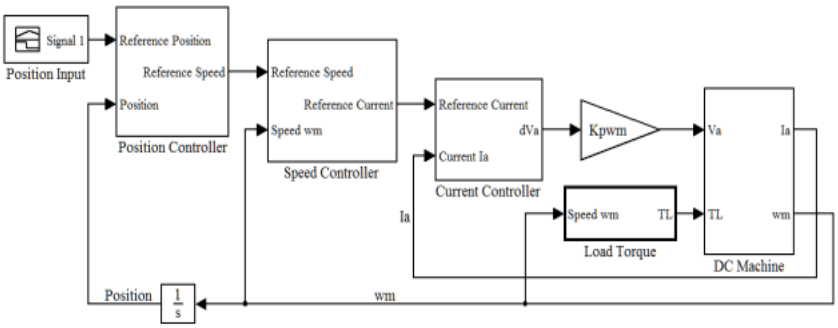


Figure 5.27: Elevator Control Model

As for the system in question, the elevator model is composed of a motor that is connected to a pulley for rotation. The pulley in turn moves the car upwards or downwards. The car is connected to the pulley via a steel wire. The specifications of the pulley radius, wire width and motor size generally follow a conventional set of regulations.

5.5.4.1.2 Platform Development and System Requirements

This lab simulation, displayed in Figure 5.28, is developed on the targeted Control application platform. In addition, the employed algorithms and libraries are based on the combination of the work shown in the Modelling section and the developed libraries for System Modelling and Control Process.

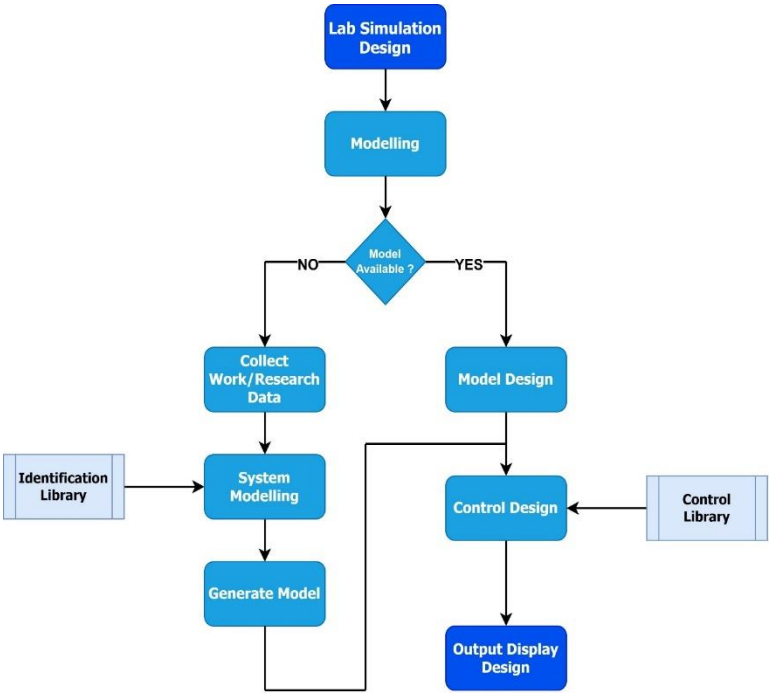


Figure 5.28: Lab Simulation Design Map

Concerning System Requirements, the integration of the developed libraries that are used for the application ensures that the virtual lab simulations can efficiently operate on the Smartphone platform. This is due to the fact that these libraries have been developed based on the previous study concerning the challenges of engineering applications on smartphones.

5.5.4.1.3 User Experience

In the elevator design, several other parameters exist for calibration. These parameters are either related to other domains such as mechanical equipment. Some of them include pulley width, counterweight, etc. Figure 5.29 shows a set of the parameters non-related to control.

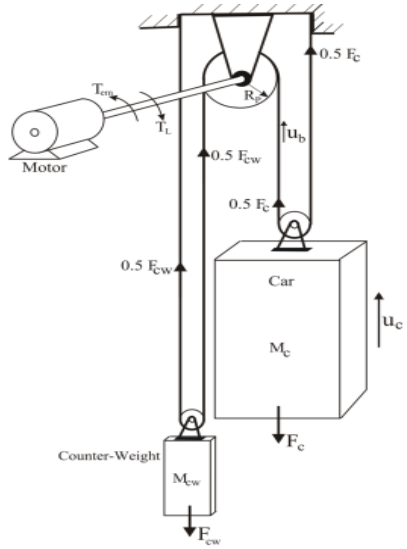


Figure 5.29: Elevator Mechanical Design

However, these parameters are based on a conventionally set design approaches. An example of this is the weight of the car, the counter-weight and the pulley radius where the relation between them is defined based on the commonly used calibrations in the industry. For these reasons, such type of parameters are previously set and the user can just focus on the calibration of the parameters related to control. Table 5.1 shows the list of parameters to be calibrated.

Elevator Control Simulation Parameters		
Highest Floor	Lowest Floor	Starting Position
Floor Sequence	Maximum Load	Cart Mass
K_P	K_I	K_D

Table 5.1: Elevator Control Simulation Parameters

The removal of the extra-parameters facilitates the User Experience as it eliminates any needed for further knowledge beyond the domain of control. In addition, an animation is added for the user to present a visual output which enhances user immersion, data clarity and in turn User Experience.

5.5.4.2 Example

The example aims to simulate the elevator control in a building. The user starts by setting up the lowest and highest floor, and afterwards, the parameters of the cart. The user is also able to modify the parameters of the PID controller that is directly connected to the system. The user sets up a floor input sequence given as 4,8,2,10. In this scenario, the starting position is at floor 0. The logic system sets up the reference sequence as 2,4,8,10. The result of the simulation is shown in Figure 5.30 where the numerical results are presented in the top graph alongside a visual animation presentation on the bottom right.

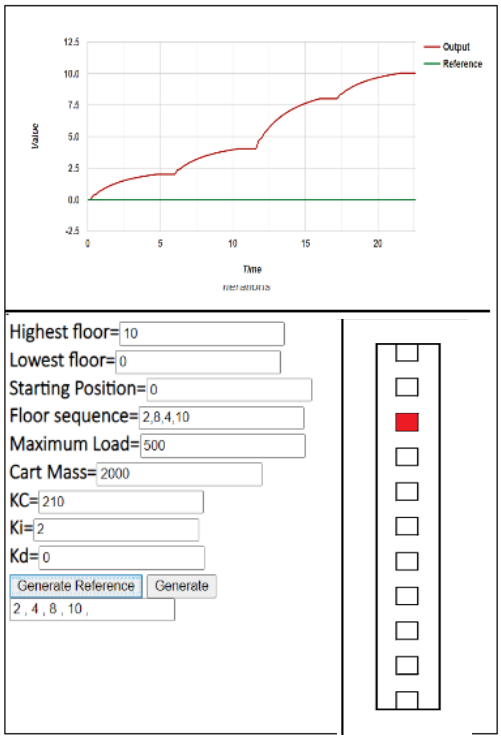


Figure 5.30: Elevator Control Example

5.6 Conclusion

This application has been developed as the primary prototype for testing proposed software and mathematical solutions. The chosen design mentality and built-in software infrastructure allows the study of the full impact of the studies as a culmination of several separate solutions. In addition, the modular design of the application allows further integration later on as more options and scenarios can be implemented. The main challenge for it remains the need of increasing control options and scenarios in order to cover a larger ground in the domain.

The analysis of the data obtained from application simulations was benchmarked with other standard engineering applications, mainly Matlab. As seen in the previous chapters, the reduction in calculation time and memory cost is considered significant. It offers a sufficient proof of concept that allows further scaling and development in the future.

Conclusion and Perspectives

The proposed challenges aimed to conceptualize and assess the obstacles related to the development of engineering applications for the smartphone platform. The conducted state-of-art shows that this concept is still in its early phase where the related applications are mostly an extension to smartphone tools or a connection to other servers and not an independent simulation application. While the reasons behind the lack of this type of application is not related to the domain of engineering, simulations in this field are considered heavy and demanding both in hardware/software. Nevertheless, the demand for it is evidently present which lead to this work. The challenges presented were explored in order to properly formulate and develop the basis of an achievable solution as seen previously. The presented solutions shows that a change in the smartphone hardware/software is not sufficient to achieve the goal but a change in the culture of thought and the basis of the developed algorithms is needed as well.

The proposed identification algorithm showed a lower mean error result in the tests that were previously in favor of the literature methods. While this method has shown success in lowering all of the observed parameters, the added liberty degrees have increased the need for manual tuning in some cases. This manual calibration can not only increase trial and error but also risk not reaching the optimal result. However, the main contribution of the proposed algorithm was in targeting the main challenges of identification algorithm implementation on smartphone platform and in particularly lowering CPU usage, mean error and maximum error.

The proposed models for LQR control define the relationship between the gain in calculation time and loss in precision. when combined with order reduction methods. This model aids in increasing the predictability in LQR calculation time to counter the known inconsistency in observed outputs. The study models the impact of matrix dimensions on the control process and in turn the implementation of methods for reducing computational complexity.

The developed application aimed for testing and validating the results. It primarily shows the design choices that contribute to the development of a control simulation application. It also offers a view of the potential of the work in increasing accessibility of complex simulation whether it is through the developed application or the presented virtual laboratories.

While the proposed software solutions offer the guide for further advancement in engineering smartphone applications, one of the current main disadvantages is the separation of any work conducted on a given engineering application from other platforms. A core feature of desktop applications is their ability to develop large scale projects that incorporate several separate applications. From a software perspective, the next direction would be developing the integration capabilities of the smartphone platform with other environments, therefore utilizing previously developed work and digital infrastructure in turn prioritizing continuation of related technology.

From a mathematical perspective, the covered processes in this work are but a small part of the abundant algorithms in Control Theory that highly vary in computational complexity. The direct next step would be assessing the impact of the proposed processes here on larger operations. One of the direct succession for the presented identification algorithm is Model Predictive Control where the focus would shift towards real-time identification. While traditional MPC relies on a pre-existing model, there are variations like Data-Driven MPC, where models are identified or updated using data collected during system operation.

In parallel, more complex processes would be targeted. A prime candidate is Linear Quadratic Gaussian control. LQG control combines LQR with a Kalman filter, which is used for state estimation in the presence of measurement noise. The Kalman filter helps to estimate the true state of the system based on noisy measurements. LQG control is also known as the Kalman-Bucy filter. This future work would come as a direct continuation of the shown models for LQR control because of the large similarity between these two types of control.

Liste de publications et participation aux conférences

[1] New Identification Algorithm for Smartphone Simulation Use

Anthony Khoury, Rachid Outbib and Rafic Younes
2023 IEEE 11th International Conference on Systems and Control (ICSC), Sousse, Tunisia, 2023, pp. 602-607, doi: 10.1109/ICSC58660.2023.10449788

[2] Challenges and Solutions for Engineering Applications on Smartphones

Anthony Khoury, Mohamad Abbas Kaddaha, Maya Saade, Rafic Younes, Rachid Outbib and Pascal Lafon
Software 2023, 2, pp. 350-376, https://doi.org/10.3390/software2030017

[3] System Control Simulation Adapted for Constrained Environment (Under preparation)

Anthony Khoury, Rachid Outbib and Rafic Younes

References

[1] M. Zaher, D. Greenwood and M. Marzouk, Mobile Augmented Reality Applications for Construction Projects, Construction Innovation, Vol. 18 No. 2, pp. 152-166, 2018.

[2] Matlab, Matlab Documentation, Matlab Help, 2022.

[3] A. Ahmadi, A. Olshevsky, P. Parrilo and J. Tsitsiklis, NP-hardness of Deciding Convexity of Quartic Polynomials and Related Problems, Mathematical Programming, Ser. A, Vol. 137, No. 1-2, pp. 453-476, 2013.

[4] D. Sha, D. Tse and J. Tsitsiklis, Hardness of Low Delay Network Scheduling, IEEE Transactions on Information Theory, Vol. 57, No. 12, pp. 7810-7818, 2011.

[5] V. Blondel and J. Tsitsiklis, A survey of computational complexity results in systems and control, Automatica Vol. 36 pp.1249-1274, 2000.

[6] V. B. Blondel, P. Koiran and J. Tsitsiklis, The Stability of Saturated Linear Dynamical Systems is Undecidable, Journal of Computer and System Sciences Vol.62, pp.442-462, 2000.

[7] M. Albreem, Approximate Matrix Inversion Methods for Massive MIMO Detectors, IEEE 23rd International Symposium on Consumer Technologies , 2019.

[8] D. Williamson and D. Shmoys, The Design of Approximation Algorithms, Cambridge University Press, 2011.

[9] L. Qian, Z. Luo, Y. Du and L. Guo, Cloud Computing: An Overview, IEEE International Conference on Cloud Computing, pp 626-631, 2009.

[10] N. Liu, P. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume and C. Maltzahn, On the role of burst buffers in leadership-class storage systems, IEEE 28th Symposium on Mass Storage Systems and Technologies pp. 1-11, 2011.

[11] J. Aughenbaugh and C. Paredis, The Role and Limitations of Modelling and Simulation in System and Designs, ASME International Mechanical Engineering Congress and RD&D Expo, 2004.

[12] M. Joorabchi, A. Mesbah and P. Kruchten, Real Challenges in Mobile App Development, ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, 2013.

[13] M. Morozov and I. Yadykin, Adaptability of Nonlinear Continuous Control Systems, Automation and Remote Control Vol.62, pp.1881-1889, 2001.

[14] J. Sneyers, T. Schrijvgers and B. Demoen, The Computational Power and Complexity of Constraint Handling Rules, ACM Transactions on Programming Languages and Systems, Vol. 31, No. 2, Article 8, 2009.

[15] J. Traub and H. Woźniakowski, Perspectives on information-based complexity, Bulletin of the American Mathematical Society, Vol.26, pp.29-52, 1992.

[16] E. Walters and S. Iden, INVENT Modeling, Simulation, Analysis and Optimization, 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 2010.

[17] J. Byrne, S. Svorobej, G. D. Tzovaras, P. Byrne, P. Östberg, A. Gourinovitch and T. Lynn, A Review of Cloud Computing Simulation Platforms and Related Environments, Proceedings of the 7th International Conference on Cloud Computing and Services Science, Vol.1, pp.679-691, 2017.

[18] A. Sulistio, C. Shin and R. Buyya, A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools, Software - Practice and Experience Issue 34, pp.653-673, 2004.

[19] K. Watson, J. Blackstone and S. Gardiner, The evolution of a management philosophy: The theory of constraints, Journal of Operations Management, Vol.25, Issue 2, pp.387-402, 2007.

[20] J. Tribbia, The Story of the World's Most Mobile Medium and How It Has Transformed Everything, Technology and Culture, Vol.47, No.3, pp. 687-690, 2006.

[21] J. Laugesen and Y. Yuan, What Factors Contributed to the Success of Apple's iPhone?, 2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), pp. 91-99, 2010.

[22] V. Lee, H. Schneider and R. Schell, Mobile Applications: Architecture, Design and Development, Association for Computing Machinery, Digital Library, 2004.

[23] S. Carvalho, M. Aniche, J. Verssimo, R. Durelli and M. Gerosa, An empirical catalog of code smells for the presentation layer of Android apps, Empirical Software Engineering Vol.24 pp. 3546–3586, 2019.

[24] K. Vishal and S. Kushwaha, Mobile Application Development Research Based on Xamarin Platform, 4th International Conference on Computing Sciences pp. 115-118, 2018.

[25] J. Swetina, G. Lu, P. Jacobs, F. Ennesser and J. Song, Toward a standardized common M2M service layer platform: Introduction to one M2M, IEEE Wireless Communications, Vol. 21, No. 3, 2014.

[26] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen and O. Spatscheck, Profiling resource usage for mobile applications: a cross-layer approach, Proceedings of the 9th

international conference on Mobile systems, applications, and services pp. 321-334, 2011.

[27] M. Hayfaa, Z. Subhi, Z. Rizgar and S. Mohammed, A Comprehensive Study of Kernel (Issues and Concepts) in different operating systems, Asian Journal of Research in Computer Science pp. 16-31, 2021.

[28] N. Chau and J. Souhwan, Dynamic analysis with Android conatiner: Challenges and opportunities, Digital Investigation Vol. 27, 2018.

[29] P. Van Overschee and D. Moor.B, Subspace Identification for Linear Systems, Theory Implementation Applications, Kluwer Academic Publishers, 1996.

[30] L. Huang and Y. Chen, A comparison between two OLS-based approaches to estimating urban multifractal parameters, Department of Geography, College of Urban and Environmental Sciences, Peking University, Beijing .

[31] J. Blais, Least Squares for Practitioners, Mathematical Problems in Engineering, Hindawi, 2010.

[32] NIST, Accounting for Non-Constant Variation across the Data, NIST Engineering Statistics Handbook, Section 4.4.5.2.

[33] D. Montgomery, E. Peck and G. Vining, Introduction to Linear Regression Analysis, Fifth Edition, Wiley Series in Probability & Statistics, 2012.

[34] S. Miller and R. Startz, Feasible Generalized Least Squares Using Machine Learning, SSRN, 2018.

[35] J. Romano and M. Wolf, Resurrecting Weighted Least Squares, Journal of Econometrics, Vol. 197, Issue 1, 2017.

[36] H. White, A Heteroskedasticity-Consistent Covariance Matrix and a Direct Test for Heteroskedasticity, Econometrica Vol. 48, pp. 817–838, 1980.

[37] J. MacKinnon and H. White, Some Heteroskedasticity-Consistent Covariance Matrix Estimators with Improved Finite Sample Properties, Journal of Econometrics. Vol. 29, pp. 305–325, 1985.

[38] F. Cribari-Neto, Asymptotic Inference Under Heteroskedasticity of Unknown Form, Computational Statistics & Data Analysis. Vol. 45, pp. 215–233, 2004.

[39] M. Kutner, C. Nachtsheim, J. Neter and W. Li, Applied Linear Statistical Models, 5th ed. New York: McGraw-Hill/Irwin, 2005.

[40] G. Ford, J. Dackson and S. Skinner, HAC standard errors and the event study methodology: a cautionary note, Applied Economics Letter Vol. 17 Issue 12, 2010.

[41] S. Bennett, Development of the PID Controller, IEEE Control Systems Magazine Vol. 13, No. 6, pp. 58-62, 1993.

[42] K. Bingi, R. Ibrahim, M. Karsiti, M. Sabo and V. Harindran, Real-Time Control of Pressure Plant Using 2DOF Fractional-Order PID Controller, Arabian Journal of Science and Engineering Vol.44, pp.2091-2102, 2018.

[43] H. Bilgic, M. Sen and M. Kalyonku, Tuning of LQR controller for an experimental inverted pendulum system based on The Bees Algorithm, Journal of Vibroengineering, Vol. 18, No. 6, pp. 3684–3694, 2016.

[44] R. Cardenas and M. Molinas, Optimal LQG Controller for Variable Speed Wind Turbine, Technoport RERC Research, Energy Procedia Vol.20, pp. 207-216, 2012.

[45] G. Da Silva, A. Bazanella, C. Lorenzini and L. Campestrini, Data-Driven LQR Control Design, IEEE Control Systems Letters Vol. 3, No. 1, pp. 180-185, 2019.

[46] H. Patino and D. Liu, Neural Network-Based Model Reference Adaptive Control, IEEE Transactions on Systems, MAN, and Cybernetics - Part B: Cybernetics Vol.30, No.1, 2000.

[47] K. Turkoglu and E. Jafarov, H inf. Loop Shaping Robust Control vs. Classical PI(D) Control: A case study on the Longitudinal Dynamics of Hezarfen UAV, Proceedings of the 2nd WSEAS International Conference on Dynamical Systems and Control, 2006.

[48] S. Quin and T. Bagdwell, A survey of industrial model predictive control technology, Control Engineering Practice Vol.11, pp. 733–764, 2003.

[49] D. Beyer, S. Lowe and P. Wendler, Reliable Benchmarking: requirements and solutions, International Journal on Software Tools for Technology Transfer Vol.21, pp.1-29, 2017.

[50] J. Salvucci and E. Chailloux, Memory consumption analysis for a functional and imperative language, Electronic notes in theoretical computer science, Electronic Notes in Theoretical Computer Science Vol.330, pp.27-46, 2016.

[51] R. Sedgewick and K. Wayne, Algorithms Fourth Edition, Princeton Education Online Databooks, 2021.

[52] F. Mala and R. Ali, The Big O Mathematics and Computer Science, Journal of Applied Mathematics and Computation Vol.6, pp.1-3, 2022.

[53] A. Krishnamoorthy and D. Menon, Matrix inversion using Cholesky decomposition, Signal Processing: Algorithms, Architectures, Arrangements, and Applications, Poznan, Poland, pp. 70-72, 2013.

[54] S. Huss-Lederman, E. Jacobson, A. Tsao, T. Turnbull and J. Johnson, Implementation of Strassen's algorithm for matrix multiplication, Proceedings on the 1996 ACM/IEEE Conference on Supercomputing, 1996.

[55] M. Lopez, J. Branke and L. Paquete, Reproducibility in Evolutionary Computation, ACM Transactions on Evolutionary Learning and Optimization Vol. 1, Issue 4 pp. 1-21, 2021.

[56] A. Boll and T. Kehrer, On the Replicability of Experimental Tool Evaluations in Model Based Development, International Conference on Systems Modelling and Management pp.111-130, 2020.

[57] P. Green, Using standards to improve the replicability and applicability of driver interface research, Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications pp.15-22, 2012.

[58] M. Stoelen, V. De Tejada, A. Huete, C. Balaguer and F. Bonsignorio, Distributed and Adaptive Shared Control Systems: Methodology for the Replication of Experiments, IEEE Robotics & Automation Magazine Vol. 22, No. 4, pp.137-146, 2015.

[59] B. Arnold, L. Bowler, S. Gibson, P. Herterich, R. Higman, A. Krystalli, A. Morley, M. Oreilly and K. Whitaker, The Turing Way: A Handbook for Reproducible Data Science, The Turing Community, 2019.

[60] A. Varga and R. Hornig, An Overview of Thee OMNeT++ Simulation Environment, 1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems, 2010.

[61] W. Bircher and L. John, Complete System Power Estimation using Processor Performance Events, IEEE Transactions on Computers Vol. 61, Issue 4 pp. 563-577, 2011.

[62] G. Cumming, Understanding the New Statistics – Effect Sizes, Confidence Intervals, and Meta-analysis, First Edition, 2012.

[63] A. Alrumayh, S. Lehman and C. Tan, Emerging mobile apps: challenges and open problems, CCF Transactions on Pervasive Computing and Interaction, Vol.3, pp.57-75, 2021.

[64] F. Kahriman and K. Liland, SelectWave: A graphical user interface for wavelength selection and spectral data analysis, Chenometrics and Intelligent Laboratory Systems Vol.212, 2021.

[65] M. Zanfardimo and R. Castaldo, MuSA: a graphical user interface for multi-OMICs data integration in radiogenomic studies, Scientific Reports volume 11, Article number: 1550, 2021.

[66] D. Carruth, C. Hudson, A. Fox and S. Deb, User Interface for an Immersive Virtual Reality Greenhouse for Training Precision Agriculture, HCII 2020: Virtual, Augmented and Mixed Reality. Industrial and Everyday Life Applications, 2020.

[67] M. Zaher, D. Greenwood and M. Marzouk, Mobile Augmented Reality Applications for Construction Projects, Construction Innovation: Information, Process, Management. Northumbria Research, 2018.

[68] G. Williams, M. Gheisari and P. Chen, An efficient BIM Translation to Mobile augmented reality applications, Journal of management in engineering Vol. 31, 2015.

[69] S. Azhar, M. Khalfan and T. Maqsood, Building Information Modeling (BIM): Now and Beyond, The Australasian Journal of Construction Economics and Building, 2012.

[70] F. Golpavar, Assessment of Collaborative Decision-Making in Design Development, UBC Thesis and Dissertation, 2006.

[71] VisualLive, Mobilive, Mixed Reality (AR) for iOS/Android, VisualLive.

[72] S. Meza, Z. Turk and M. Dolenc, Measuring the potential of augmented reality in civil engineering, Advances in Engineering Software Vol. 90, 2015.

[73] A. Shakil, A Review on Using Opportunities of Augmented Reality and Virtual Reality in Construction Project Management, OTMC International Journal, 2019.

[74] Y. Kim and W. Kim, Implementation of Augmented Reality System for Smartphone Advertisement, International Journal of Multimedia and Ubiquitous Engineering Vol.9, No.2, 2014.

[75] P. Yoora, S. Hyojoo and K. Changwan, Investigating the determinants of construction professionals' acceptance of web-based training: An extension of the technology acceptance model, Automation in construction Vol.22, 2012.

[76] Z. Kanetaki, C. Stergiou, G. Bekas and E. Kanetaki, Machine Learning and Statistical Analysis applied on Mechanical Engineering CAD course: A Case Study During ERTE Pahse in the Context of Higher Education, 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies, 2020.

[77] E. Modic, Linear Bearing CAD models on the fly, Today's Medical Development, 2018.

[78] A. Nazar, P. Jiao, Q. Zhang, K. Egbe and A. Alavi, A new structural health monitoring approach based on smartphone measurements of magnetic field intensity, IEEE Instrumentation and Measurement magazine Vol. 24, No. 4, pp. 49-58, 2021.

[79] D. H. Park, J. M. Heo, W. Jeong, Y. H. Yoo and B. J. Park, Smartphone-Based VOC Sensor Using Colorimetric Polydiacetylenes, ACS Applied Material Interfaces, 2018.

[80] A. Minichiello, D. Armijo, S. Mukherjee, L. Caldwell, V. Kulyukin, T. Truscott, J. Elliott et A. Bhouraskar, Developing a mobile application-based particle image velocimetry tool for enhanced teaching and learning in fluid mechanics: A design-based research approach, Computer Applications in Engineering Education Vol.29, Issue 10, 2020.

[81] M. Tiago De Oliveira, M. Maija, M. Leslie, L. Terhi, T. Sarang and L. Ville, Towards tropospheric delay estimation using GNSS smartphone receiver network, Advances in Space Research Vol.68, Issue 12, 2021.

[82] R. Alturki and V. Gay, Usability attributes for mobile applications, a systematic review, Faculty of Engineering and Information Technology, University of Technology Sidney, 2018.

[83] L. Punchoojit, Usability Studies on Mobile User Interface Design Patterns: A Systematic Literature Review, Advances in Human-Computer Interactions, Hindawi, 2017.

[84] K. Bennett, A. Nagy and J. Flash, Visual display, Handbook of Human Factors and Ergonomics, 2012.

[85] A. Abdalha and A. Muasaad, A study of the interface usability issues of mobile learning applications for smartphones from the user's perspective, International Journal on Integrating Technology in Education Vol.3, No.4, 2014.

[86] A. Oulasvirta, P. Kristenson, X. Bi and A. Howes, Computational Interaction, Oxford University Press, 2017.

[87] J. Nichols, D. Chau and B. Myers, Demonstrating the viability of automatically generated user interfaces, Proc SIGCHI Conf. Human Factors Comput System, pp. 1283-1292, 2007.

[88] C. Zeidler, G. Weber, W. Stuerzlinger and C. Lutteroth, Automatic generation of user interface layouts for alternative screen orientations, Proc. IFIP Conf. Human-Comput. Interact, pp. 13-35, 2017.

[89] A. Oulasvirta, N. Dayama, M. Shiripour, M. John and M. Karrenbauer, Combinatorial Optimization of Graphical User Interface Designs, Proceedings of the IEEE, Vol. 108, 2020.

[90] E. Deelman and A. Chervenak, Data Management Challenges of Data-Intensive Scientific Workflows, Eighth IEEE International Symposium on Cluster Computing and the Grid , 2008.

[91] J. Sosa, P. Thomas, L. Delia and J. Caseres, Mobile Application Development Approaches: A Comparative Analysis on the Use of Storage Space, Argentine Congress of Computer Science XXIV, 2018.

[92] K. Vandenbrouke, D. Ferreira and J. Goncalvez, Mobile cloud storage: a contextual experience, Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services, 2014.

[93] C. Li, J. Bai and J. Tang, Joint optimization of data placement and scheduling for improving user experience in edge computing, Journal of Parallel and Distributed Computing, 2019.

[94] A. Carroll and G. Hesier, An analysis of power consumption in a smarpthone, USENIX annual technical conference, 2010.

[95] L. Kaufman, Data Security in the World of Cloud Computing, IEEE Security & Privacy Vol. 7, No.4, pp. 61-64, 2009.

[96] W. Lyu, J. Gui, M. Wan and M. Halfond, An Empirical Study of Local Database Usage in Android Applications, IEEE International Conference on Software Maintenance and Evolution , 2017.

[97] T. Wang, S. Oral, Y. Wang, B. Settlemyer, S. Atchley and W. Yu, BurstMem: A High-Performance Burst Buffer System for Scientific Applications, IEEE International Conference on Big Data, pp. 71-79, 2014.

[98] S. Thapaliya, P. Bangalore, J. Lofstead, K. Mohror and A. Moody, Managing I/O Interference in a Shared Burst Buffer System, 45th International Conference on Parallel Processing, pp. 416-425, 2016.

[99] T. Want, K. Mohror, A. Moody, K. Sato and W. Yu, An Ephemeral Burst-Buffer File System for Scientific Applications, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2016.

[100] M. Ataeeshojai, R. Elliott, W. Krzymien, C. Tellambura and I. Maljevic, Iterative Matrix Inversion Methods for Precoding in Cell-Free Massive MIMO Systems, IEEE Transactions on Vehicular Technology, 2022.

[101] M. Albreem, Approximate Matrix Inversion Methods for Massive MIMO Detectors, IEEE 23rd International Symposium on Consumer Technologies , 2019.

[102] A. Tarek, A generalized set theoretic approach for time and space complexity analysis of algorithms and functions, Proceedings of the 10th WSEAS International Conference on Applied Mathematics, 2006.

[103] J. Gao, A. Sivaraman, A. Argawal, H. Li and L. Peh, DIPLOMA: Consistent and coherent shared memory over mobile phones, IEEE 30th International Conference on Computer Design, 2012.

[104] K. Cooper and T. Harvey, Compiler-controlled memory, ACM SIGOPS Operating Systems Review Vol.32, 1998.

[105] P. Ou and B. Desmky, Towards understanding the costs of avoiding out-of-thin-air results, Proceedings of the ACM on Programming Languages Vol.2, 2018.

[106] S. Gotz, M. Tichy and T. Kehrer, Claimed advantages and disadvantages of (dedicated) model transformation languages: a systematic literature review, Software & Systems Modeling, 2021.

[107] S. Gotz, M. Tichy and T. Kehrer, Dedicated Model Transformation Languages vs. General-purpose Languages: A Historical Perspective on ATL vs. Java, Science & Technology Publications, 2021.

[108] S. Hopner, T. Kehrer and M. Tichy, , Contrasting dedicated model transformation languages versus general purpose languages: a historical perspective on ATL versus Java based on complexity and size, Software & Systems Modeling, 2022.

[109] J. Sjoberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P. Glorennec, H. Hjalmarsson and A. Juditsky, Nonlinear black-box modeling in system identification: a unified overview, Automatica, Vol.31, Issue 12, pp.1691-1724, 1995.

[110] G. Hanrahan and D. Patil, Chemometrics & Statistics; Multivariate Calibration Techniques, Encyclopedia of Analytical Science (Second Edition), 2005.

[111] I. Steinwart, D. Hush and C. Scovel, Optimal Rates for Regularized Least Squares Regression, Los Alamos National Laboratory, 2009.

[112] D. Maggin, H. Swaminathan, H. Rogers, B. O'keeffe, G. Sugai and R. Horner, A generalized least squares regression approach for computing effect sizes in single-case research: Application examples, Journal of School Psychology, 2011.

[113] J. Gulich, Centrifugal Pumps, Library of Congress Control Number: 2007934043, 2008.

[114] S. Horn, R. Horn and D. Duncan, Estimating heteroskedastic variances in linear models, Journal of the American Statistical Association Vol.70, pp.380-385, 1975.

[115] Y. Yuan and D. MacKinnon, Robust Mediation Analysis Based on Median Regression, Psychology Methods, National Center for Biotechnology Information, 2014.

[116] P. Yalavarthy, B. Pogue, H. Dehghani and K. Paulsen, Weight-matrix structured regularization provides optimal generalized least-squares estimate in diffuse optical tomography, International Journal of Medical Physics Research and Practice, Vol. 34, Issue 6, Part 1, pp. 2085-2098, 2007.

[117] E. Andries and J. Kalivas, Interrelationships between generalized Tikhonov regularization, generalized net analyte signal, and generalized least squares for desensitizing a multivariate calibration to interferences, Journal of Chenometrics , 2013.

[118] K. Hirose and H. Masuda, Robust Relative Error Estimation, National Center for Biotechnology Information , 2018.

[119] C. Kat and P. Els, Validation metric based on relative error, Taylor & Francis, Mathematical and Computer Modelling of Dynamical Systems Vol. 18, No. 5, pp.487-520, 2012.

[120] D. Kothari and I. Nagrath, Theory and Problems of Basic Electrical Engineering, 2nd Edition, PHI Learning Private Limited, Delhi, 2016.

[121] V. Naik, D. Sonawane, D. Ingole and D. Ginoya, Model Predictive Control of DC Servomotor using Active Set Method, IEEE International Conference on Control Applications, 2013.

[122] J. Karki, Active Low-Pass Filter Design, Texas Instruments, AAP Precision Analog, SLOA049D, 2023.

[123] V. Adetola and M. Guay, Performance Improvement in Adaptive Control of Linearly Parameterized Nonlinear Systems, IEEE Transactions on Automatic Control, vol. 55, no. 9, pp. 2182-2186, 2010.

[124] G. Da Silva, A. Bazanella, C. Lorenzini and L. Campestrini, Data-Driven LQR Control Design, IEEE Control Systems Letters, vol. 3, no. 1, pp. 180-185, 2019.

[125] A. Laub, A Schur method for solving algebraic Riccati equations, IEEE Transactions on Automatic Control Vol. 24, No. 6, pp. 913-921, 1979.

[126] C. C., The convergence analysis of an accelerated iteration for solving algebraic Riccati equations, Journal of the Franklin Institute, Vol. 359. Issue 1, pp. 619-636, 2022.

[127] V. Simoncini, D. Szyld and M. Monsalve, On two numerical methods for the solution of large-scale algebraic Riccati equations, IMA Journal of Numerical Analysis, Vol. 34, Issue 3, pp. 904–920, 2014.

[128] M. Opmeer, Decay of Hankel singular values of analytic control systems, Systems & Control Letters, Vol.59, Issue 10, pp. 635-638, 2010.

[129] Y. Liu and B. Anderson, Singular Perturbation Approximation of Balanced Systems, Proceedings of the 28th Conference on Decision and Control, 1989.

[130] R. Walker and A. Ryson, Reliable Algorithm for Modal Decomposition, Journal of Guidance, Control and Dynamics Vol.13, No.5, pp.893-904, 1990.

[131] R. Schafer, What Is a Savitzky-Golay Filter?, IEEE Signal Processing Magazine pp.111-116, 2022.

[132] J. Guinon, E. Ortega, J. Garcia and V. Perez, Moving Average and Savitzki-Golay Smoothing Filters Using Mathcad, International Conference on Engineering Education, 2007.

[133] R. Zanetti, Recursive Update Filtering for Nonlinear Estimation, IEEE Transactions on Automatic Control Vol. 57, No. 6, pp. 1481-1490, 2012.

[134] F. Hampel, A general qualitative definition of robustness, The Annals of Mathematical Statistics, Vol.42, No.6, pp.1887-1896, 1971.

[135] H. Frey and S. Patil, Identification and Review of Sensitivity Analysis Methods, Risk Analysis Vol.22, No. 3, 2002.

[136] A. Somani, P. Chakrabarti and A. Patra, An Evolutionary Algorithm-Based Approach to Automated Design of Analog and RF Circuits Using Adaptive Normalized Cost Functions, IEEE Transactions on Evolutionary Computation, Vol. 11, No. 3, 2007.

[137] E. Dougherty, S. Kim and Y. Chen, Coefficient of determination in nonlinear signal processing, Signal Processing, Vol. 80, Issue 10, pp. 2219-2235, 2000.

[138] H. Kordestani, Zhang, S. CMasri and M. Shadabfar, An empirical time-domain trend line-based bridge signal decomposing algorithm using Savitzky–Golay filter, Journal of The International Association for Structural Control and Monitoring, 2021.

[139] R. Dolbeau, Theoretical peak FLOPS per instruction set: a tutorial, Journal of Supercomputing vol.74, pp. 1341-1377, 2017.

[140] M. Blaeser, Fast Matrix Multiplication, Theory of Computing Library Graduate Surveys Issue 5, 2013.

[141] A. Antoulas, D. Sorensen and Y. Zhou, On the decay rate ofHankel singular values and related issues, Systems and Control Letters Vol.46, pp.323-342, 2002.

[142] R. Opmeer, Decay of Hankel singular values of analytic control systems, Systems and control letters, vol.59 (2010) no.10, pp 635-638, 2010.

[143] S. Gugercin and C. Antoulas, A comparative study of 7 algorithms for model reduction, Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, NSW, Australia, pp. 2367-2372 vol.3, 2000.

[144] D. Enns, Model Reduction with Balanced Realizations: An Error Bound and a Frequency Weighted Generalization, Proceedings of the 23rd Conference on Decision and Control pp. 127-132, 1984.

[145] J. Stigter, L. Willigenburg and J. Molenaar, An Efficient Method to Assess Local Controllability and Observability for Non-Linear Systems, International Federation on Automatic Control Vol.51, Issue 2 pp.535-540, 2018.

[146] P. Massioni, H. Raynaud, C. Kulcsar and J. Conan, An Approximation of the Riccati Equation in Large-Scale Systems With Application to Adaptive Optics, IEEE Transactions on Control Systems Technology Vol. 23, No. 2, pp. 479-487, 2015.

[147] J. Beckers, Adaptive Optics for Astronomy: Principles, Performance, and Applications, Annual Review of Astronomy and Astrophysics Vol. 31 pp.13-62, 1993.

[148] C. Petit, J. Conan, C. Kulcsar, H. Raynaud, T. Fusco, J. Montri and D. Rabaud, Optimal control for Multi-Conjugate Adaptive Optics, Comptes Rendus Physique Vol.6, Issue 10 pp.1059-1069, 2005.

[149] C. Petit, J. Conan, C. Kulcsar and H. Raynaud, Linear quadratic Gaussian control for adaptive optics and multiconjugate adaptive optics: experimental and numerical analysis, Journal of the Optical Society of America A Vol. 26, Issue 6, pp. 1307-1325, 2009.

[150] C. Kulcsar, H. Raynaud, C. Petit and J. Conan, Minimum variance prediction and control for adaptive optics, Automatica Vol.48 Issue 9 pp.1939-1954, 2012.

[151] V. Lakshminarayanan and A. Fleck, Zernike polynomials: a guide, Journal of Modern Optics Vol. 58, No. 7 pp.545-561, 2011.

[152] B. Le Roux, J. Conan, C. Kulcsar, H. Raynaud, L. Mugnier and T. Fusco, Optimal control law for classical and multiconjugate adaptive optics, Journal of the Optical Society of America A Vol. 21, Issue 7, pp. 1261-1276, 2004.

[153] K. Ismara, A. Suharjono and D. Supriadi, Ubiquitous learning in occupational health and safety for vocational education, International Journal of Evaluation and Research in Education, 2021.

[154] G. Moustefaoui and F. Tariq, Mobile Apps Engineering: Design, Development, Security, and Testing, CRC Press, 2018.

[155] N. Reid and S. Iqbal, The role of laboratory work in university chemistry, Chemical Education Research & Practice, 2005.

[156] U. Kanli and R. Yagbasan, The Effects of a Laboratory Approaches on the Development of University Students' Science Process Skills and Conceptual Achievement, Essays in Education, Vol.24, Issue 1, 2008.

[157] H. Schmidgen, History of the Beginnings of the Laboratory in the Early Modern World, Journal of European History Online, 2011.

[158] R. Trumper, The Physics Laboratory – A Historical Overviewand Future Perspectives, Journal of Science & Education, 2003.

[159] L. Fiesel and A. Rosa, The Role of the Laboratory in undergraduate engineering application, Journal of Engineering Education , 2005.

[160] Z. Bishop, T. Howard, P. Lazari, B. Taylor, P. Trend and A. Funnel, Student Experiences of Practical Activities During the Covid-19 Pandemic, White Rose Research Online, 2021.

[161] O. Zine, M. Errouha, M. Zamzounm, A. Derouish and A. Talbi, A costless and effective remote measurement laboratory in electrical engineering, International Journal of Electrical Engineering & Education, 2019.

[162] E. Faulconer and G. Amy, A Review to Weigh the Pros and Cons of Online, Remote, and Distance Science Laboratory Experiences, The International Review of Research in Open and Distributed Learning, 2018.

[163] E. Giannaka, A. Alexio and C. Bouras, Virtual Laboratories in Education, International Federation for Processing Digital Library; Technology Enhanced Learning, 2005.

[164] D. Olivotti, S. Dreyer, P. Kolsch, C. Herder, M. Breitner and J. Aurich, Realizing availability-oriented business models in the capital goods industry, 10th CIRP Conference on Industrial Product-Service Systems, 2018.

[165] P. Khatri, Getting Started with Simulink in Matlab: Designing a Model, Circuit Digest, 2018.

[166] C. Pearson and C. Kudzai, Virtual Laborаties: A Solution For Tertiary Science Education in Bostwana, European Journal of Logistics, Purchasing and Supply Chain Management Vol.3, No.3, pp.12-24, 2015.

[167] R. Schekler, Virtual labs: A substitute for traditional labs?, The International Journal Of Developmental Biology, 2003.

[168] A. Shamshin, Detailed analysis of the advantages of virtual laboratory work in physics, Journal "ScienceRise: Pedagogical Education", 2021.

[169] R. Oser, Effectiveness of Virtual Laboratories in Terms of Achievement, Attitudes, and Learning Environment among High School Science Students, Curtin University, Science and Mathematics Education Centre, 2013.

[170] G. Eri, A. Antonios and C. Bouras, Virtual Laboratories in Education, International Federation for Information Processing Digital Library; Technology Enhanced Learning, 2005.

[171] A. Al Mousawi, A. Ambusaidi, S. Al-Balushi and K. Al-Balushi, The impact of using virtual lab learning experiences on 9th grade students' achievement and their attitudes towards science and learning by virtual lab, Journal of Turkish Science Education, 2018.

[172] B. Celine, F. Nsanganwimama and A. Tarmo, Effectiveness of Virtual Laboratories in Teaching and Learning Biology: A Review of Literature, International Journal of Learning, Teaching and Educational Research, 2022.

[173] S. Hopner, M. Tichy and T. Kehrer, Contrasting dedicated model transformation languages versus general purpose languages:a historical perspective on ATL vs java based based on complexity and size, Software and systems modeling 2022, 2022.

[174] A. Tarek, A generalized set theoretic approach for time and space complexity analysis of algorithms and functions, Proceedings of the 10th WSEAS International Conference on APPLIED MATHEMATICS, Dallas, Texas, USA, 2006.

[175] G. Iyer and K. Kalyandurgmath, The study of methods and challenges faced by science teachers while conducting practical (lab) online in Mumbai region due to covid 19, International journal of creative research thoughts, 2021.

[176] S. De Las Heras, B. Kensington-Miller, B. Young, V. Gonzalez, U. Kruhne, S. Mansouri and S. Baroutian, Benefits and Challenges of a Virtual Laboratory in Chemical and Biochemical Engineering: Students' Experiences in Fermentation, ACS Publications, 2021.

[177] N. Arun, S. Velmurugan and A. Ashir, Development of a Mathematical Model Using Statistical Design of Experiments for the Determination of Gas Holdup in Inverse-Fluidized Bed, Mathematical Modelling and Applications to Industrial Problems pp.497-503, 2011.

[178] H. El-Sabbagh, The Impact of a Web-Based Virtual Lab on the Development of Students'Conceptual Understanding and Science Process Skills, Dresden University of Technology, 2010.

[179] M. Barak and Y. Dori, Enhancing undergraduate students' chemistry understanding through project-based learning in an IT environment, Journal of Science Education, 2004.

[180] R. Robles, Simulink model of the X-15's Adaptive Flight Control System, Mathworks, 2021.

[181] N. Tamura, H. Taira and K. Umemura, Challenges of a "Virtual" Biophysics Laboratory during the "Stay home"period caused by COVID-19, Journal of Physics: Conference Series, 2021.

[182] L. Ngoyi, Benefits and Challenges Associated with Using Virtual Laboratories and solutions to overcome them, Virginia Polytechnic Institute and State University, 2013.

[183] N. Al-Saiyd and E. Zriqat, Analyzing the Impact of Requirement Changing on Software Design, European Journal of Scientific Education , 2015.

[184] J. Faias, J. Ferreira and C. Leite, Accessibility and Usability of Online Labs: Real barriers in a virtual world, Conference: 4th International Conference on Remote Engineering and Virtual Instrumentation, 2007.

[185] A. Carroll and G. Heiser, An Analysis of Power Consumption in a Smartphone, Proceedings of the 2010 USENIX conference on USENIX annual technical conference, 2010.

[186] J. Wu and A. Bose, A new successive relaxation scheme for the W-matrix solution method on a shared memory parallel computer, IEEE Transactions on Power Systems, 1996.

[187] M. Adnan, Z. Sickder, M. Mushfiquzzoha and M. Zulfikar, A Simulation Study of an Elevator Control system using digital logic, International Journal of Engineering Trends and Technology, 2017.

[188] P. Ford, Electric elevator drive with position control, Louisiana State University and Agricultural and Mechanical College, 2012.