An Interval Approach for Stability Analysis

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Science and Engineering

2019

Eduard A. Codres

School of Engineering
Department of Electrical and Electronic Engineering
Contents
List of figures ......................................................................................................................... 7
List of acronyms ...................................................................................................................... 10
Abstract .................................................................................................................................. 11
Declaration ................................................................................................................................. 13
Copyright ................................................................................................................................. 15
Acknowledgements .................................................................................................................. 17
Chapter 1: Introduction ......................................................................................................... 19
  1.1 Motivation ....................................................................................................................... 19
  1.2 Stability challenges ......................................................................................................... 21
  1.3 Problem statement ......................................................................................................... 22
  1.4 Contributions .................................................................................................................. 23
  1.5 Thesis organisation ......................................................................................................... 24
Chapter 2: Nonlinear dynamical systems - stability analysis from an interval perspective ....... 25
  2.1 Nonlinear dynamical systems ......................................................................................... 25
  2.2 Stability for nonlinear dynamical systems ..................................................................... 26
  2.3 Interval methods ............................................................................................................. 28
    2.3.1 Introduction ............................................................................................................... 28
    2.3.2 Intervals ................................................................................................................... 28
    2.3.3 Contractors ............................................................................................................. 30
    2.3.4 Separators .............................................................................................................. 32
    2.3.5 Subpavings ............................................................................................................. 32
    2.3.6 Set inversion ........................................................................................................... 32
  2.4 Eulerian methods based on Lyapunov theory. An interval approach.............................. 35
  2.5 Lagrangian methods based on interval simulation .......................................................... 38
  2.6 Problems in interval simulations ................................................................................... 39
    2.6.1 Range evaluation of an interval function ................................................................. 39
    2.6.2 Wrapping effect ..................................................................................................... 40
  2.7 Approaches using regions ................................................................................................ 41
  2.8 Approaches using real trajectories ................................................................................ 43
  2.9 Conclusions .................................................................................................................... 43
Chapter 3: Sliding mode control of a differential drive robot .................................................. 44
  3.1 Introduction ..................................................................................................................... 44
    3.1.1 Sliding mode fault tolerant control for mobile robots .............................................. 44
## List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Unstable dynamical systems. a) Snow avalanche; b) Forest fire</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>Illustrative representation of Lagrangian and Eulerian methods</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>The exact solution $\mathbf{X}$ is represented by the grey area</td>
<td>34</td>
</tr>
<tr>
<td>4</td>
<td>The outer approximation box $\mathbf{X}$</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>The solution is refined using the SIVIA algorithm with separators</td>
<td>34</td>
</tr>
<tr>
<td>6</td>
<td>A trajectory starting in (-1.5,-1.5) has $V(x_1,x_2) = 0$ every time $x_2 = 0$.</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>V-Stability for $V(\mathbf{x}) \in [0,\infty)$</td>
<td>37</td>
</tr>
<tr>
<td>8</td>
<td>Wrapping effect</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>Non-holonomic robot</td>
<td>46</td>
</tr>
<tr>
<td>10</td>
<td>Diagram of the WMR</td>
<td>49</td>
</tr>
<tr>
<td>11</td>
<td>Comparison between the simulated robot model and the real robot</td>
<td>57</td>
</tr>
<tr>
<td>12</td>
<td>Robot position and velocities with a fault in the right wheel</td>
<td>58</td>
</tr>
<tr>
<td>13</td>
<td>Trajectory tracking parameters</td>
<td>60</td>
</tr>
<tr>
<td>14</td>
<td>Block diagram of the control structure with the kinematic SMC and PI controllers</td>
<td>62</td>
</tr>
<tr>
<td>15</td>
<td>Trajectory tracking results for the kinematic SMC tested in simulation</td>
<td>63</td>
</tr>
<tr>
<td>16</td>
<td>Trajectory tracking results for the kinematic SMC applied to the real robot</td>
<td>65</td>
</tr>
<tr>
<td>17</td>
<td>Block diagram of the control structure with the Kinematic+Dynamic SMC and PI controllers</td>
<td>68</td>
</tr>
<tr>
<td>18</td>
<td>Test results when the robot has no fault</td>
<td>69</td>
</tr>
<tr>
<td>19</td>
<td>Test results when the robot has a fault (flat tyre)</td>
<td>70</td>
</tr>
<tr>
<td>20</td>
<td>Test results when the robot has a fault and $\Delta R=0$</td>
<td>71</td>
</tr>
<tr>
<td>21</td>
<td>Comparison between the angular velocities of the wheels</td>
<td>72</td>
</tr>
<tr>
<td>22</td>
<td>Test results when using torque control for the dynamic SMC in simulation</td>
<td>73</td>
</tr>
<tr>
<td>23</td>
<td>Test results when using torque control for the dynamic SMC with the real robot</td>
<td>73</td>
</tr>
<tr>
<td>24</td>
<td>Block diagram of the kinematic SMC with dynamic SMC and torque control</td>
<td>75</td>
</tr>
<tr>
<td>25</td>
<td>Simulation results for the kinematic SMC with separate dynamic SMC</td>
<td>76</td>
</tr>
<tr>
<td>26</td>
<td>The control signals</td>
<td>77</td>
</tr>
<tr>
<td>27</td>
<td>The control surfaces do not converge to zero</td>
<td>77</td>
</tr>
<tr>
<td>28</td>
<td>Block diagram of the kinematic SMC with dynamic SMC and current SMC for the DC motors</td>
<td>79</td>
</tr>
<tr>
<td>29</td>
<td>Test results when the robot has a fault in the left wheel (flat tyre)</td>
<td>80</td>
</tr>
<tr>
<td>30</td>
<td>Control torques generated by the dynamic SMC and armature current for each DC motor</td>
<td>81</td>
</tr>
<tr>
<td>31</td>
<td>Fault hiding for the sliding mode fault-tolerant control of a nonholonomic mobile robot for fault in the actuators (flat tyre)</td>
<td>82</td>
</tr>
<tr>
<td>32</td>
<td>Simulation results of trajectory tracking using an SMC based on a kinematic model in a non-faulty situation</td>
<td>85</td>
</tr>
<tr>
<td>33</td>
<td>Simulation results of trajectory tracking using an SMC based on a kinematic model in a faulty situation without fault hiding block</td>
<td>87</td>
</tr>
<tr>
<td>34</td>
<td>Simulation results of trajectory tracking using an SMC based on a kinematic model in a faulty situation with the fault hiding block</td>
<td>89</td>
</tr>
<tr>
<td>35</td>
<td>Sivia for $V(s) &lt; 0$</td>
<td>93</td>
</tr>
<tr>
<td>36</td>
<td>Sivia for $V(s) &lt; 0$</td>
<td>94</td>
</tr>
<tr>
<td>37</td>
<td>Sivia for $V(s) &lt; 0$, with two different sets of parameters</td>
<td>95</td>
</tr>
<tr>
<td>38</td>
<td>Sivia for $V(s) &lt; 0$, with two different sets of parameters</td>
<td>98</td>
</tr>
<tr>
<td>39</td>
<td>Pendulum diagram</td>
<td>101</td>
</tr>
<tr>
<td>40</td>
<td>Paving for $g(x_1,x_2) &lt; 0$. $g(x_1,x_2)$ is zero for $x \in [\pm \pi/2,0]$</td>
<td>102</td>
</tr>
<tr>
<td>41</td>
<td>Cross-out condition evaluated in $\mathbf{X}$boundary</td>
<td>102</td>
</tr>
</tbody>
</table>
The new $g_P(x_1,x_2) = 0$ with the perturbations added becomes a positive invariant set for the system with uncertainty.

Figure 47. Paving for $g(x_1,x_2) < 0$. $g(x_1,x_2)$ is zero for $x \in [\pm 0.8464,0]$.  

Figure 48. Cross-out condition evaluated when $g(x_1,x_2) = 0$.  

Figure 49. $g_P(x_1,x_2)$ is negative on the border and $g_P(x_1,x_2) = 0$ becomes an invariant set. 

Figure 50. Cross-out condition evaluated when $g_P(x_1,x_2) = 0$ with perturbation ($\varepsilon = 0.1, \lambda = 0.4$).  

Figure 51. Cross-out condition for $g(x_1,x_2) = 0$, $g(x_1,x_2)$ is positive in two regions.  

Figure 52. Cross-out condition for $g_P(x_1,x_2) = 0$ with perturbations ($\varepsilon = 0.52, \lambda = 0.445$).  

Figure 53. $g_P(x_1,x_2)$ is negative on the border and $g_P(x_1,x_2) < 0$.  

Figure 54. Paving for $V(x_1,x_2) < 1$. The cross-out condition is negative on the border.  

Figure 55. Plot for the set $V(x_1,x_2) \geq 0$.  

Figure 56. Plot for $V(x_1,x_2)$ on the border of $V(x_1,x_2) < 0.1$. $V(x_1,x_2)$ is positive.  

Figure 57. $V(x_1,x_2) < 0.25$ is the smallest invariant set for the system with uncertainty.  

Figure 58. Plot for the biggest invariant set that does not include (0,0).  

Figure 59. Paving of the feasible solution set for the system without uncertainty.  

Figure 60. Paving of the feasible solution set with better accuracy.  

Figure 61. Paving of the feasible solution set for the system with additive uncertainty $[-0.05,0.05]$.  

Figure 62. Paving of the feasible solution set for the system with additive uncertainty $[-0.1,0.1]$.  

Figure 63. Paving for $g_P(x_1,x_2), \varepsilon, \lambda = 0$ with $\varepsilon = [0,2.5]$ and $\lambda = [0,2.5]$.  

Figure 64. Paving of the feasible solution set for the system without uncertainty.  

Figure 65. Paving of the feasible solution set for the system with additive uncertainty $[-0.05,0.05]$.  

Figure 66. Paving of the feasible solution set for the system with additive uncertainty $[-0.1,0.1]$.  

Figure 67. Paving for $g_P(x_1,x_2), \varepsilon, \lambda = 0$ with $\varepsilon = [0,0.5]$ and $\lambda = [0,0.5]$.  

Figure 68. Paving of the feasible solution set for $(\varepsilon, \lambda)$ when the system energy is approximated with a circle.  

Figure 69. Positive invariant set obtained with perturbation parameters: $\varepsilon = 0.5, \lambda = 1$.  

Figure 70. Paving of the feasible solution set for $(\varepsilon, \lambda)$ when the system energy is approximated with an ellipse.  

Figure 71. Positive invariant set obtained with perturbation parameters: $\varepsilon = 0.36, \lambda = 67$.  

Figure 72. Lyapunov function plot for the time-variant pendulum system (t=0.57).  

Figure 73. Lyapunov function plot for the time variant pendulum system (t=0.628).  

Figure 74. Illustration of the Picard-Lindelöf operator for the tube $[x](\dot{t})$.  

Figure 75. Representation of an RBFN network.  

Figure 76. Weights evolution during training.  

Figure 77. Output error evolution during training.  

Figure 78. Sine function approximated with an RBFN with 15 basis functions.  

Figure 79. Vector field evaluation: \textbf{a)} Interval evaluation; \textbf{b)} Vertex evaluation.  

Figure 80. Vertex propagation for the trajectories starting in the initial set.  

Figure 81. Two RBFNs, $\psi^{OUT}$ and $\psi^{IN}$, are initialised and trained.  

Figure 82. Inner and outer approximations of the capture tube: $\psi^{OUT}$ is yellow and $\psi^{IN}$ is green.  

Figure 83. Training error.  

Figure 84. Paving of the initial set and outbound trajectories.  

Figure 85. Outbound trajectories on the surface of the initial box for two regions of the state space.
Figure 86. Euler Vertex propagation for the initial set..........................................................138
Figure 87. Convex hull of the vertex propagation.................................................................139
Figure 88. Non-Convex hull of the vertex propagation............................................................139
Figure 89. SIVIA generated paving of a simplex (the simplex is triangle in 3D)......................141
Figure 90. Mesh accuracy control :a) Simplex bisection; b) Simplex union...........................142
Figure 91. Capture tube bracketing for the pendulum system..................................................146
Figure 92. Inner and outer mesh approximations: .................................................................147
Figure 93. Different perspective for the inner and outer approximations of the capture tube. ..........147
Figure 94. A propagation in a more nonlinear region of the dynamical system and with longer propagation time.................................................................148
Figure 95. Propagation with an equilibrium point inside the initial box.................................148
Figure 96. Propagation finishes when all trajectories return to the initial box...........................149
Figure 97. Comparison between RBFN propagation (left) and linear interpolation propagation (right)....150
Figure 98. Comparison with Dynibex for the system without parametric uncertainty..................150
Figure 99. Comparison with Dynibex for the system with parametric uncertainty.....................151
Figure 100. Comparison with Dynibex for a third order system...............................................152
Figure 101. Comparison with Dynibex when the initial box is located in a more complex region of the dynamical system.................................................................152
## List of acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODE</td>
<td>Ordinary Differential Equation</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>RBFN</td>
<td>Radial Basis Function Network</td>
</tr>
<tr>
<td>LPV</td>
<td>Linear Parameter Varying</td>
</tr>
<tr>
<td>CSP</td>
<td>Constraint Satisfaction Problem</td>
</tr>
<tr>
<td>SIVIA</td>
<td>Set Inversion Via Interval Analysis</td>
</tr>
<tr>
<td>FTC</td>
<td>Fault Tolerant Control</td>
</tr>
<tr>
<td>SMC</td>
<td>Sliding mode control</td>
</tr>
<tr>
<td>WMR</td>
<td>Wheeled Mobile Robot</td>
</tr>
</tbody>
</table>
Abstract

An Interval Approach for Stability Analysis

Eduard Alexandru Codres
A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2019

The stability problem plays an important role not only in industry but also in our everyday life. Unstable systems are dangerous and can cause financial losses or can lead to injuries and loss of human life. Starting with the 19th century, many methods were proposed for analysing the stability of linear dynamical systems, nonlinear dynamical systems, and for systems with uncertainties.

This thesis proposes several methods to study the stability of nonlinear dynamic systems with uncertainties. The majority of existing methods can only be applied to nonlinear dynamical systems under some particular circumstances. New approaches are developed in this thesis by combining the existing theory with interval analysis. The new methods are used to analyse the stability of Sliding Mode Control (SMC) strategy, for generating automatic Lyapunov functions and for solving Ordinary Differential Equations (ODE) and ODE with uncertainties.

New types of sliding mode control structures, able to cope with faults, are developed and tested on a differential drive robot. Suitable tuning parameters for the SMC when dealing with multiple uncertainties are found by using novel methods which employ interval analysis techniques. To the best of my knowledge, there is no analytical method for tuning and estimating the robustness for an SMC in the presence of uncertainties.

Moreover, the stability of uncertain nonlinear dynamical systems was studied from a Lyapunov perspective. Lyapunov functions can be automatically generated for mechanical systems based on their energy. In the majority of the cases, the energy function does not guarantee an asymptotic convergence to the equilibrium point. LaSalle theorem solved this problem for systems without uncertainties. However, when the systems have uncertainty it is not possible to apply LaSalle invariant principle. In this thesis, this was solved by adding perturbation functions to the energy function of the system, and by using interval analysis, the asymptotic convergence for systems with uncertainties was proved.

New guaranteed integration methods are proposed for solving nonlinear ODEs in the last part of the thesis. It was shown, that by using interval methods, we are able to bracket the boundary of the solution manifold. Two methods for generating a boundary approximation were proposed: the first method used radial basis function networks and the second method used linear interpolation for a mesh triangulation.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DoculInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University's policy on presentation of Theses.
Acknowledgements

I would like to thank my academic supervisor, Dr. Alexandru Stancu, for his continuous help and guidance during my research. I would also like to thank my colleagues Mohamed Mustafa, Salvador Pacheco and Mario Martinez for the great talks and debates we had.

My research would not have been possible without the financial support of The Aerospace Research Institute of the University of Manchester. In this regard, I would like to express my gratitude towards Professor Constantinos Soutis, the Director of the Aerospace Research Institute, for his trust in my abilities.

I dedicate this work, firstly, to my family, my wife and children, for being next to me and coping with the difficulties of moving to a new country.

I also dedicate this work to my parents, my mother and my late father, who nurtured my curiosity and my independence.

Lastly, I would like to express my gratitude towards late Professor Gheorghe Puscasu, who helped me forge my path and develop my academic and professional abilities.
Chapter 1: Introduction

1.1 Motivation

A dynamical system can be described by a state equation which represents a differential equation of the form \( \dot{x}(t) = f(x(t)) \) also called Ordinary Differential Equation (ODE). All objects and phenomena around us whose states are in a continuous change in time can be described by a differential equation. A very important concept related with the differential equations (dynamic systems) is the stability. Stability analysis of dynamical systems has been receiving a lot of attention since the appearance of humans on Earth. At the beginning, unstable systems like a snow avalanche (Figure 1a), a forest fire (Figure 1b), etc. were related with the Gods. The humans thought that these unstable phenomena were created by Gods as revenge, when the humans did not obey them. However, stable dynamical systems were used by humans since their early existence to help them in many everyday activities, such as using a wheel, a carriage, archery, etc. It took many centuries until the humanity was able to understand these behaviours and was able to explain and describe them in a scientific and rigorous way, sometimes by developing new mathematical tools.

![Avalanche and Fire](image)

*Figure 1. Unstable dynamical systems. a) Snow avalanche; b) Forest fire*

The stability of a given state of motion was first studied in the 17th Century by one of the greatest minds of all time, the English mathematician, astronomer and physicist, Sir Isaac Newton [1]. He used his new theory to study the motion of the Moon around the Earth. Sir Isaac Newton was the first researcher who understood that the masses possess energy. Newton has understood that it is easier to study the rate of change (the derivative of the position of a moving object) instead of its position in space, i.e. the trajectory. This concept leads to the foundation of calculus, a new mathematical domain necessary to describe the movement. Newton’s theory was later developed...
even further by mathematicians and astronomers in their efforts to study the movement of celestial bodies in order to prove that their motion is in accordance with Newton’s principle of universal gravitation.

Calculus and Newton’s results constituted the foundation for the stability analysis of the dynamic systems when, two centuries later (in the 19th century), a Russian mathematician, Aleksandr Mikhailovich Lyapunov, has introduced the most complete work regarding the stability analysis of dynamical systems [2]. It was when the necessary conditions for a dynamical system to be stable were formulated. His direct method [3] [4] [5] is based on finding a function of the state coordinates for a dynamical system, named Lyapunov function, which is positive-definite and its rate of change around the equilibrium point is always negative or zero. This result is very intuitive, i.e., a dynamic system is stable if its energy is always decreasing. If a Lyapunov function exists, then the system is said to be stable. Lyapunov’s method only provides the necessary condition and not the sufficient one, i.e. if it is not possible to find a Lyapunov function for a dynamic system, then it is not possible to state that the system is not stable.

Lyapunov theory addresses two important stability concepts: asymptotic stability for a dynamic system when the rate of change of the Lyapunov function is always negative and Lyapunov stability for a dynamic system when the rate of change of the Lyapunov function is zero. In the first case, when the derivative of the Lyapunov function is negative definite, the asymptotic convergence of the system trajectory towards the equilibrium point is proved. In the second case, when the derivative of the Lyapunov function is just negative semidefinite, then the solution will converge to the set:

\[ \mathbb{A} = \{ \mathbf{X} | \dot{V}(\mathbf{X}) = 0 \} \]

where \( V(\mathbf{X}) \) is the Lyapunov function.

Now, \( \mathbb{A} \) is pessimistic, (i.e., it contains states toward which the system cannot converge) and is too large to help us find a significant limit set. In many engineering applications, Lyapunov stability is not enough to prove the asymptotic convergence of the systems states towards the equilibrium points. For example, when a pendulum is disturbed from its nominal position, we not only want the pendulum to maintain its position in a range given by the magnitude of the disturbance (Lyapunov stability) but also require that the pendulum position gradually go back to its original position (asymptotic stability). Because even in most trivial stability problems the derivative of the Lyapunov function is just negative semidefinite then it is not possible to prove the asymptotic stability. For the pendulum example it is intuitive that after applying the disturbance, the pendulum will go back to its
initial position in finite time, the Lyapunov function related with the energy is negative semidefinite. So, using Lyapunov theory we cannot conclude that the pendulum will go back to its stable equilibrium point. We need something additional to be used in conjunction with Lyapunov theory to be able to prove the asymptotic stability. This was an important, yet unsolved, problem which concerned the researchers for another century. However, this problem was solved in the 20th century, by LaSalle [6] who has extended Lyapunov stability theory proposing his famous theorem. The theorem relaxed the conditions of Lyapunov theorem to work for the case when the derivative of the Lyapunov function is just negative semidefinite. According to LaSalle theorem, the set $\mathcal{A}$ is contracted because the system will converge to the largest subset $\mathcal{A}$ of $\mathcal{A}$ which is positive invariant. Hence, if the dynamic system has only one equilibrium point inside the set $\mathcal{A}$, then the trajectories of the dynamic system will always converge to that equilibrium point which is the largest invariant subset of set $\mathcal{A}$. If the system does not have any equilibrium points inside the set $\mathcal{A}$, then the trajectories of the dynamic systems will converge to a periodic orbit.

LaSalle-Barbashin-Krasovskii invariance principle [7] [8] is now used together with Lyapunov’s theory to prove the asymptotic stability for autonomous dynamical systems $\dot{x} = f(x)$. This theory was also extended later for time-variant dynamical systems $\dot{x} = f(x, t)$ by LaSalle [9] [10] and Artstein [11] [12].

Almost another century passed until the researchers were able to find mathematical tools, such as Interval Methods [13], to solve the differential equations with uncertainties. The uncertainties can be located in the initial states, can be additive or multiplicative (located in the system parameters). During the last decades of the 20th century and in the first decades of the 21st century, the researchers have addressed the problem of dynamical systems with uncertainties and several methods were proposed to study their stability. In this case, the solution of the differential equation is not a singular trajectory but, instead, a set of trajectories which are the solution of a differential inclusion, i.e., $\dot{x} \in F(x)$. Lyapunov theory cannot be applied directly in this case.

1.2 Stability challenges

Some of the main problems, yet challenges, encountered in the stability research area and addressed in this thesis are listed below.

**Challenge 1. To extend LaSalle invariant principle for dynamic systems with uncertainties.** For general mechanical systems, we can obtain an analytical expression for the Lyapunov function
but we have no method to automatically compute \( \mathbb{A} \) (the largest positive invariant set). In general, the energy functions are available for mechanical systems and they can be considered as Lyapunov candidates. However, in the majority of the cases, the derivatives of the energy functions are only negative semidefinite. Moreover, when the dynamic system is with uncertainty, it is not possible to directly apply LaSalle theorem. There exists the necessity to introduce a new method in order to prove the asymptotic stability for the solution of a differential inclusion.

**Challenge 2. A new method to tune the parameters for the Sliding Mode Controller.** The Lyapunov theory was not addressed seriously by the research community before the control systems became an important domain in many industrial applications. One of the big problems existing in control systems research area in the early 1950s was the lack of methods to prove the stability of the new proposed control strategies. For instance, adaptive control was a very promising control strategy to design aircraft autopilots, but the adaptive control was soon abandoned due to the crash of the test flight. Nonlinear control strategies gained importance once again at the end of the 1970s when Lyapunov methods were used more and more to prove the stability for the new control methods. However, for some important nonlinear control strategies, such as Structure Variable Control methods and its well-known control strategy, Sliding Mode, it does not yet exist any analytical or numerical method to rigorously tune the controller parameters. This is a very important problem since for a robust control strategy we cannot quantify how big its robustness is.

**Challenge 3. Computing the solutions for ODE with uncertainties.** There is a lack of general methods to solve ODEs with uncertainties. There are many methods proposed in the literature which can find the solutions of an ODE with uncertainties (differential inclusion) only for some particular cases. However, all these methods are affected by several problems, such as wrapping effect [14] and computational cost for high dimensions [15]. These problems make them suitable only for isotonic systems [16]. There is a need for a general method to approximate the solutions for ODEs with uncertainties suitable for high dimensional problems.

### 1.3 Problem statement

This thesis attempts to solve the stability problem for nonlinear systems with uncertainties using interval methods. Interval methods are computational methods that can perform, in a natural way, nonlinear computations with sets of real numbers. They are at the core of guaranteed system solving methods that can prove the existence of a solution and, if the latter is not unique, compute
the set of all solutions while taking into account all sources of uncertainty. These methods have direct applicability to a broad range of scientific areas from engineering, to financial and medical domains and they proved to be very efficient when applied in the stability research area.

As it was stated in the previous paragraph, it is difficult to find an analytical and closed form solution for the stability problem for dynamic systems with uncertainties. In this thesis, the problem of stability is solved using efficient and guaranteed numerical search methods. In this way, we can provide an approximated solution. The following questions will be addressed and several methods will be proposed as possible answers:

- How to apply interval methods to solve the LaSalle theorem for dynamical systems with uncertainties (Challenge 1).
- How to tune the parameters for the sliding mode controller to have a stable controller for a dynamic system with uncertainties (Challenge 2).
- How to employ interval methods to propose a general method to solve the integration of ODEs with uncertainties (Challenge 3).

1.4 Contributions

The contributions made in this thesis are threefold:

1. To develop a general method to extend the LaSalle theorem for systems with uncertainties. A new method to compute the largest positive invariant subset $\bar{\mathbb{A}}$ of the set $\mathbb{A}$ is proposed.

2. To provide a method which allows tuning the Sliding Mode Control parameters for dynamic systems with uncertainties. This is a very important result which can be used to quantify the robustness of this control method, needed when Sliding Mode is used as a fault-tolerant control strategy.

3. Bracketing the manifolds which contain the solutions of an ODE with uncertainties (differential inclusion). To achieve this, two methods are proposed in this thesis. The first method uses Radial Basis Function Networks (RBFN) to approximate the boundary of the solution manifold. To the best of my knowledge, this is the first attempt in which a neural network is trained using intervals to solve the stability problem for dynamic systems with uncertainties. The second method is based on using a mesh triangulation and linear interpolation to approximate the boundary of the solution manifold and to obtain the inner and the outer brackets.
1.5 Thesis organisation

The outline of this thesis is as follows:

Chapter 2 describes the stability problem and presents the state-of-the-art in stability research domain. Interval analysis and interval computations are also introduced in this chapter. Moreover, it is illustrated how interval methods can be employed efficiently to solve nonlinear problems.

Chapter 3 deals with novel tuning methods for the parameters of Sliding Mode Control applied for dynamic systems with uncertainties. In Chapter 4 the LaSalle method is extended for dynamic systems with uncertainties. A novel method based on perturbation theory and interval methods is proposed and illustrated using case studies. In Chapter 5 two methods for bracketing the manifold of the solutions of an ODE with uncertainty were proposed.

Finally, Chapter 6 presents conclusions and reflections about the results proposed in the thesis, as well as some comments about future work.
Chapter 2: Nonlinear dynamical systems - stability analysis from an interval perspective

2.1 Nonlinear dynamical systems

Dynamical system theory allows us to model and study the behaviour of phenomena that undergoes spatial and temporal evolution. In general, a dynamical system can be modelled using three elements:

- *state space* – the space in which the dynamical behaviour takes place;
- *dynamic* – specifies the evolution of the system over time;
- *initial condition* – the initial starting state of the system.

Dynamical systems exist in every field (science, engineering, medicine, economics, etc.) and differential equations are used to analytically represent them. With the three elements enumerated above and the external input to the dynamical system, we can generate the following differential equation to model it:

\[ \dot{x}(t) = f(t, x(t), u(t)), \quad x(t_0) = x_0, \quad t \in [t_0, t_1] \]

where:

- \( x(t) \in \mathcal{D}, \quad t \in [t_0, t_1] \), is the dynamical state or the trajectory of the system and \( \mathcal{D} \) is an open subset of \( \mathbb{R}^n \) with \( 0 \in \mathcal{D} \);
- \( u(t) \in U \subseteq \mathbb{R}^m \) is the input or the control signal;
- \( f: [t_0, t_1] \times \mathcal{D} \times U \rightarrow \mathbb{R}^n \) is a nonlinear function, piecewise continuous in \( t \) and continuous in \( x \) and \( u \).

The system described equation (2.1) is also called a nonlinear disturbed or controlled time-varying dynamical system. If \( u(t) \equiv 0 \) then the system becomes:

\[ \dot{x}(t) = f(t, x(t)), \quad x(t_0) = x_0, \quad t \in [t_0, t_1], \]

In this case, the system is called a *nonlinear undisturbed* or *uncontrolled time-varying* dynamical system. If \( f(t, x) = f(t_0, x) \) for all \( (t, x) \in [t_0, t_1] \times \mathcal{D} \) then the system is called a *time invariant* or *autonomous* dynamical system.

For a nonlinear dynamical system a point \( x_e \in \mathcal{D} \) is said to be an *equilibrium point* if at time \( t_e \in [t_0, t_\infty] \) we have \( f(t, x_e) = 0 \) for all \( t \geq t_e \).
2.2 Stability for nonlinear dynamical systems

One of the most important aspects in system theory is stability analysis of dynamical systems. An equilibrium point of a dynamical system is said to be stable if, when disturbed from its nominal position, the perturbed motion remains in an arbitrarily small region of the state space. If all trajectories of the dynamical system approach the equilibrium point for large values of time, then the system is said to be asymptotically stable.

Generally speaking, to prove that the dynamic system (2.1) is stable can be done in two ways: 1) by verifying that the solution of (2.1) is stable (converges to the equilibrium point) or 2) by checking that the region of the state space where the solution lies is bounded. These two methods lead to two important groups of approaches [17], embracing all algorithms existing in the literature, i.e., Lagrangian approach and Eulerian approach. This classification was proposed for the first time in a paper coming from the fluid mechanics research area [18]. In order to describe both approaches, the author in [18] has proposed an interesting experiment where an observer moves or not with the fluid parcel. To better understand these concepts, we will give an intuitive example considering the wind as fluid (see Figure 2). The wind flag indicator represents the Eulerian perspective and the flag represents the vector field and describes the wind behaviour. This approach is similar to Lyapunov based approaches. The second approach, the Lagrangian approach, is represented by the leaves which flow together with the wind. This is an alternative way to describe the wind dynamic which can be interpreted as the leaves moving with the wind. In other words, in Eulerian approach, the observer does not move with the dynamic system but, instead, is observing how the state space is influenced by the dynamical system. In contrast, in Lagrangian approach, the observer moves with it. Hence, in conclusion, the observer can be viewed as an estimation of the system trajectories.

![Figure 2. Illustrative representation of Lagrangian and Eulerian methods.](image-url)
Numerical methods such as interval analysis can be successfully employed for both approaches, Eulerian and Lagrangian. The Lagrangian method was used most of the times in conjunction with interval analysis to deal with uncertainties in nonlinear dynamical systems.

The Lagrangian methods are related to the guaranteed integration approaches which are extensively addressed by researchers coming from several research areas. In guaranteed integration, the idea is to compute the smallest interval containing all possible solutions (trajectories).

When we deal with a dynamical system such as (2.1), the evolution function \( f(t, x(t)) \) is the speed of the fluid and the state \( x(t) \) is the position of a fluid parcel at time \( t \). For a Lagrangian based approach, we have to first define a target and then perform simulations to find the states that reach the target [19]. Most of the existing methods found in the literature are designed for linear dynamic systems or they can be used if the system is nonlinear after a linearisation is applied [17] or if the nonlinear system can be represented as an Linear Parameter Varaing (LPV) system.

In opposition to the Lagrangian methods, the Eulerian approach was mainly addressed by Jaulin in [20] and by Jaulin and Stancu in [21] [22] and the main idea is to avoid the integration of the ODE. In these papers, V-Stability is introduced as a stability method for dynamic systems with uncertainties. Any method which does not require integration through time can be considered to belong to the Eulerian category of methods. This is why Lyapunov-based methods [23] [24] or level-set methods [25] can also be considered as Eulerian. These methods only require checking some constraints in the state space. However, in order to be able to check the constraints, we need to know the parametric expression for Lyapunov functions [26].

In the case of Lagrangian methods, the problem of guaranteed integration has been studied extensively by researchers from different fields of research and several algorithms have been proposed:

- **Qualitative Reasoning** [27] [28] [29];
- **Constraint Satisfaction** [30];
- **Validated Initial Value Problems** [31] [32] [33] [14] [34];
- **Automatic Control** [35] [36] [37] [38] [39];
- **Fault Detection** [40] [41] [42] [43] [44];
- **Circuit Tolerance Analysis** [45] [46];
- **Computer Science** [15];
All these algorithms can be classified according to the method used to compute the interval hull of the set of estimated states. There are two main approaches in the literature:

- one-step-ahead iteration based on previous approximations of the set of estimated states;
- a set of punctual trajectories generated by selecting particular values from the afferent interval using heuristics or optimisation.

2.3 Interval methods

2.3.1 Introduction

Interval methods are computational methods that can perform, in a natural way, nonlinear computations with sets of real numbers [13]. This can be used for solving mathematical problems using computers in almost any field which involves them.

The basic concept that defines the foundation for interval methods is to enclose real numbers with intervals and vectors with boxes. By extending the algebra of the real numbers to its interval counterpart, it was first possible to implement guaranteed classical numerical algorithms. Interval methods were developed further to include algorithms which can deal with sets and solve mathematical problems that, in this case, do not have real counterparts.

2.3.2 Intervals

A real interval \([x]\) is a connected, closed subset of \(\mathbb{R}\), and it is denoted by \([x] = [\underline{x}, \overline{x}]\) where \(\underline{x} \in \mathbb{R}\) and \(\overline{x} \in \mathbb{R}\) are the lower and upper bounds, respectively.

The lower bound of an interval is defined as:

\[
\underline{x} = \sup\{a \in \mathbb{R} \cup (-\infty, \infty) \mid \forall x \in [x], a \leq x\},
\]

whilst the upper bound of an interval is defined as:

\[
\overline{x} = \inf\{b \in \mathbb{R} \cup (-\infty, \infty) \mid \forall x \in [x], x \leq b\}\]

Operations defined for sets can also be used for intervals. The intersection of two intervals \([x]\) and \([y]\) is an interval and it is defined as:

\[
[x] \cap [y] = \{z \in \mathbb{R} \mid z \in [x] \text{ and } z \in [y]\}\]

The union of two intervals is not always an interval:

\[
[x] \cup [y] = \{z \in \mathbb{R} \mid z \in [x] \text{ or } z \in [y]\}\]
The set of all real intervals of $\mathbb{R}$ is denoted by $\mathbb{I}\mathbb{R}$. In order to make $\mathbb{I}\mathbb{R}$ closed with respect to union we have to define the *interval hull* of a subset $\mathbb{X}$ of $\mathbb{R}$ as the smallest interval $[\mathbb{X}]$ that contains it. In this way, we can define the interval union as:

$$[x] \cup [y] = [[x] \cup [y]]$$

Example:

$$[1,2] \cup [3,4] = [1,4]$$

Basic arithmetic operations can be applied to intervals. Define $\circ$ as a binary operator and $\circ \in \{+, -, *, /\}$, then:

$$[x] \circ [y] = [[x \circ y \in \mathbb{R} \mid x \in [x], y \in [y]]]$$

Examples:

$$[-1,2] + [0,1] = [-1,3]$$
$$[-1,2] \times [2,3] = [-3,6]$$
$$\frac{[-1,2]}{[1,2]} = [-1,2]$$

A function $f : \mathbb{R} \to \mathbb{R}$ can be extended to intervals as follows:

$$[f](\mathbb{X}) = \{f(x) \mid x \in \mathbb{X}\}$$

$[f](\mathbb{X})$ is an *inclusion function* which satisfies $f([\mathbb{X}]) \in [f](\mathbb{X})$. The evaluation of an interval function can be pessimistic, meaning that the result can be bigger than it is supposed to be for the given intervals. One of the reasons is the multi-occurrence for some of the function’s variables.

For continuous elementary functions, we have $f([\mathbb{X}]) = [f(\mathbb{X})]$.

Examples:

$$\text{sqr}([-2,5]) = [0,25]$$
$$\text{sqr}(1,4) = [1,2]$$
$$\text{sqr}([-5,-4]) = \emptyset$$
$$\text{sqr}([-5,1]) = \text{sqr}(0,1) = [0,1]$$
$$\text{exp}(0,1) = [1, e]$$
$$\text{sin}(0, \pi) = [0,1]$$

In order to extend intervals to higher dimensions interval vectors are defined as the Cartesian product of closed intervals. An interval vector $[\mathbb{x}]$, also called a *box*, can be written as:

$$[\mathbb{x}] = [x_1] \times [x_2] \times \ldots \times [x_n]$$

where:

$$[x_1], [x_2], \ldots [x_n] \in \mathbb{I}\mathbb{R},$$
$$[\mathbb{x}] \in \mathbb{I}\mathbb{R}^n,$$
$$\mathbb{I}\mathbb{R}^n- \text{ is the set of all n-dimensional boxes.}$$
2.3.3 Contractors

A constraint is a relation between a number of variables \( x_i \in \mathbb{R}, i \in \{1, \ldots, n\} \) such as:
\[
f_1(x_1, x_2, \ldots, x_n) = 0, \\
f_2(x_1, x_2, \ldots, x_n) \geq 0.
\]

Constraints expressed by inequalities can be converted to equalities by adding an extra variable such as:
\[
f_2(x_1, x_2, \ldots, x_n) \geq 0 \iff f_2(x_1, x_2, \ldots, x_n) + x_{n+1} = 0, x_{n+1} \in (-\infty, 0]
\]

A constraint satisfaction problem (CSP) [13] is defined by a set of variables \( x_i \in \mathbb{R}, i \in \{1, \ldots, n\} \) and a set of constraints \( f_j, j \in \{1, \ldots, m\} \):
\[
f_j(x_1, x_2, \ldots, x_n) = 0
\]

In vector form: \( f(\mathbf{x}) = 0, f: \mathbb{R}^n \to \mathbb{R}^m, \mathbf{x} = (x_1, x_2, \ldots, x_n)^T. \)

For the CSP to be completely defined each variable \( x_i \) has to belong to a domain \( 
\mathbb{X}_i \). In our case, for simplicity, the domains will be considered intervals \( [x_i] \). Then the CSP is formulated as:
\[
\mathcal{H}: (f(\mathbf{x}) = 0, \mathbf{x} \in [\mathbf{x}])
\]

The solution set for the above CSP consists of all the feasible values of \( [x_i] \) which satisfy all constraints and it is defined as:
\[
\mathcal{S} = \{\mathbf{x} \in [\mathbf{x}] \mid f(\mathbf{x}) = 0\}
\]

A contractor [13] \( C \) is an operator applied to a CSP that computes a subset \( [\mathbf{x}'] \) of the initial domain \( [\mathbf{x}] \) such that \( \mathcal{S} \subset [\mathbf{x}'] \subset [\mathbf{x}] \).

In other words, a contractor is an operator from \( \mathbb{R}^n \) to \( \mathbb{R}^n \) that satisfies the following properties:
- **Contractance**: \( C([\mathbf{x}]) \subset [\mathbf{x}] \);
- **Consistency**: \( C([\mathbf{x}]) \cap \mathcal{S} = [\mathbf{x}] \cap \mathcal{S} \) (\( \mathcal{S} \) is consistent with \( C \); \( \mathcal{S} \sim C \));
- **Monotonicity**: \( [\mathbf{x}] \subset [\mathbf{y}] \Rightarrow C([\mathbf{x}]) \subset C([\mathbf{y}]) \).

A contractor \( C \) is **minimal** for a CSP if for any other contractor \( C_1 \) associated with the same CSP, we have \( C \subset C_1 \). The minimal contractor computes the smallest box \( [\mathbf{x}] \) that includes the solution set \( \mathcal{S} \).

Many types of contractors can be defined for various types of CSPs. Some of them are derived from the classical point algorithms by extending them to intervals (Gauss elimination, Gauss-Seidel or Newton algorithms [13]). The forward-backward [13] contractor uses constraint propagation to contract the initial domain and it can be applied to a wide range of CSPs. One of the advantages of this
contractor is that it can deal with each constraint of the CSP individually, thus the number of constraints can be very big compared to the number of variables.

As an example, we can consider the constraint $x_3 = x_1 - 2x_2$ and the box $[x] = [1,10] \times [1,10] \times [1,10]$. This constraint can be written in three ways:

$$x_1 = x_3 + 2x_2,$$
$$x_2 = \frac{x_1 - x_3}{2},$$
$$x_3 = x_1 - 2x_2.$$

By applying interval computations on the above equations we get:

$$([x_3] + 2[x_2]) \cap [x_1] = ([1,10] + 2 \times [1,10]) \cap [1,10] = [3,30] \cap [1,10] = [3,10]$$
$$\frac{[x_1] - [x_3]}{2} \cap [x_2] = \frac{[3,10] - [1,10]}{2} \cap [1,10] = [\frac{-7.9}{2}] \cap [1,10] = [-3.5,4.5] \cap [1,10] = [1.45]$$
$$([x_1] - 2[x_2]) \cap [x_3] = ([3,10] - 2 \times [1.45]) \cap [1,10] = [-6.8] \cap [1,10] = [1.8]$$

If we apply the contractor again the initial domain $[x]$ would not be contracted anymore. The forward-backward contractor is minimal for this example, but for other types of constraints (with nonlinearities, with multi-occurrence for some of the variables) it is not minimal. In this case, the contractor can be applied repeatedly until the result converges to the solution $[x]_\infty$. In practice, the algorithm is stopped if the difference between two successive contractions is smaller than a given threshold e.g. $\frac{\text{width}(x_{k+1})}{\text{width}(x_k)} < \varepsilon$ for $\varepsilon > 0$. If a contractor has this convergence property it is called a fixed-point contractor.

A more general algorithm for the forward-backward contractor $C_{\leftrightarrow}$ was proposed in [13] where each constraint of the CSP $f_i(x_1, x_2, ..., x_n) = 0$ is decomposed into primitive constraints using elementary operations and functions ($+$, $-$, $\times$, $/$, $\sin$, $\cos$, etc.).

<table>
<thead>
<tr>
<th>Algorithm $C_{\leftrightarrow} (in:f, \varepsilon; out:[x])$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. do</td>
</tr>
<tr>
<td>2. $[x_k] = [x]$;</td>
</tr>
<tr>
<td>3. $[y] \equiv [f([x_k]) \cap {0}]$;</td>
</tr>
<tr>
<td>4. $[x] \equiv [y] \cap [f^{-1}([x_k])]$;</td>
</tr>
<tr>
<td>5. while $\frac{\text{width}(x_{k+1})}{\text{width}(x_k)} &gt; \varepsilon$</td>
</tr>
<tr>
<td>6. return $[x]$;</td>
</tr>
</tbody>
</table>
2.3.4 Separators
A separator \([47]\) \(S\) is a pair of contractors \(\{S^\text{in}, S^\text{out}\}\) such that
\[
S^\text{in}(\{x\}) \cup S^\text{out}(\{x\}) = [x] \quad \text{(complementarity)},
\]
for all \([x] \in \mathbb{I}^n\).
A set \(S\) is consistent with the separator \(S\) (we will write \(S \sim S\)), if:
\[
S \sim S^\text{out} \quad \text{and} \quad \overline{S} \sim S^\text{in}, \quad \text{where} \quad \overline{S} = \{x | x \notin S\}.
\]
The remainder of a separator \(S\) is defined as:
\[
\partial S(\{x\}) = S^\text{in}(\{x\}) \cap S^\text{out}(\{x\}).
\]

2.3.5 Subpavings
As presented above, intervals and boxes can be used to approximate sets with the advantage of being easy to represent and manipulate in a computer. However, sometimes a better approximation of a given set is needed. In this case, we can choose to cover the set with a union of subsets of \(\mathbb{R}^n\) which are easy to represent and manipulate (boxes, ellipsoids, zonotopes, polytopes etc.).

For a set of interest \(X\) some important properties can be proven by using such a covering. Further, we will use sets of non-overlapping boxes of \(\mathbb{R}^n\) to cover the set \(X\), also called subpavings. To prove some properties for \(X\) we can also bracket the set using inner and outer approximations \((X, \overline{X})\) such that:
\[
X \subset X \subset \overline{X}
\]
A subpaving of a box \([X] \subset \mathbb{R}^n\) is a union of non-overlapping subboxes of \([X]\) with non-zero width. A subpaving of \([X]\) is regular if each of its boxes can be obtained from \([X]\) by a finite succession of bisections and selections. Regular subpavings, also called \(n\)-trees, are easier to manipulate with a computer.

2.3.6 Set inversion
For any function \(f: \mathbb{R}^n \rightarrow \mathbb{R}^m\) and any subset \(Y\) of \(\mathbb{R}^m\), the set inversion is the set \(X\), where:
\[
X = \{x \in \mathbb{R}^n | f(X) \in Y\} = f^{-1}(Y)
\]
Two regular subpavings \((\underline{X}, \overline{X})\) \(\underline{X} \subset X \subset \overline{X}\) can be found using SIVIA algorithm (Set Inversion Via Interval Analysis [13]). The basic SIVIA algorithm uses \([f](\cdot)\) (a convergent inclusion function for \(f\)) and bisections to find the two bracket regular subpavings of \(X\). The solution is refined using bisections until a given accuracy \(\varepsilon\) is reached. A more efficient version of SIVIA can be obtained by using contractors together with bisections in order to improve the accuracy for the same number of bisections. The only drawback in this case is that the inner and outer approximations are not regular subpavings anymore as it was shown by Jaulin in [13].
The basic SIVIA algorithm (without contractors) is presented in the table below:

<table>
<thead>
<tr>
<th>Algorithm SIVIA ( (in: f, Y, [X], \varepsilon; \text{inout: } \mathbb{X}, \mathbb{X} ) ) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. if ( <a href="%7BX%7D">f</a> \cap Y = \emptyset )</td>
</tr>
<tr>
<td>2. return;</td>
</tr>
<tr>
<td>3. if ( <a href="%7BX%7D">f</a> \subset Y )</td>
</tr>
<tr>
<td>4. ( \mathbb{X} := \mathbb{X} \cup {X} )</td>
</tr>
<tr>
<td>5. ( \mathbb{X} := \mathbb{X} \cup {X} )</td>
</tr>
<tr>
<td>6. return;</td>
</tr>
<tr>
<td>7. if ( \text{width}({X}) &lt; \varepsilon )</td>
</tr>
<tr>
<td>8. ( \mathbb{X} := \mathbb{X} \cup {X} )</td>
</tr>
<tr>
<td>9. else</td>
</tr>
<tr>
<td>10. ( ([X_1], [X_2]) := \text{bisection}(X) )</td>
</tr>
<tr>
<td>11. SIVIA( (f, Y, [X_1], \varepsilon \mathbb{X}, \mathbb{X}) )</td>
</tr>
<tr>
<td>12. SIVIA( (f, Y, [X_2], \varepsilon \mathbb{X}, \mathbb{X}) )</td>
</tr>
<tr>
<td>13. return;</td>
</tr>
</tbody>
</table>

The SIVIA algorithm using separators \( S = \{ S_{\text{in}}, S_{\text{out}} \} \) is presented in the table below:

<table>
<thead>
<tr>
<th>Algorithm SIVIA_( S ) ( (in: S, [X], \varepsilon; \text{inout: } \mathbb{X}, \mathbb{X}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( { [X_{\text{in}}], [X_{\text{out}}] } = S({X}) )</td>
</tr>
<tr>
<td>2. ( \mathbb{X} := \mathbb{X} \cup {X} \setminus [X_{\text{in}}] )</td>
</tr>
<tr>
<td>3. ( \mathbb{X} := \mathbb{X} \cup {X} \setminus [X_{\text{in}}] )</td>
</tr>
<tr>
<td>4. ( [X] = [X_{\text{in}}] \cap [X_{\text{out}}] )</td>
</tr>
<tr>
<td>5. if ( \text{width}({X}) &lt; \varepsilon )</td>
</tr>
<tr>
<td>6. ( \mathbb{X} := \mathbb{X} \cup {X} )</td>
</tr>
<tr>
<td>7. else</td>
</tr>
<tr>
<td>8. ( ([X_1], [X_2]) := \text{bisection}(X) )</td>
</tr>
<tr>
<td>9. SIVIA( (S, [X_1], \varepsilon \mathbb{X}, \mathbb{X}) )</td>
</tr>
<tr>
<td>10. SIVIA( (S, [X_2], \varepsilon \mathbb{X}, \mathbb{X}) )</td>
</tr>
<tr>
<td>11. return;</td>
</tr>
</tbody>
</table>

**Example.** Define the following constraint satisfaction problem:

\[
\begin{align*}
x_2 &\geq x_1^2 \\
x_2 &\leq \frac{1}{2} x_1^2 + 1
\end{align*}
\]

This can be cast into a set inversion problem if we define a function \( f: \mathbb{R}^2 \rightarrow \mathbb{R}^2 \):

\[
f(X) = \begin{pmatrix} x_1^2 - x_2 \\ \frac{1}{2} x_1^2 - x_2 + 1 \end{pmatrix}
\]

The solution set for this CSP is:

\[
\mathbb{X} = f^{-1}([{-\infty, 0}] \times [0, \infty])
\]
Figure 3. The exact solution $\mathcal{X}$ is represented by the grey area.

Figure 4. The outer approximation box $\overline{\mathcal{X}}$ (yellow box) is obtained after the forward-backward contractor $\mathcal{C}_{\Rightarrow}$ defined for this CSP is applied to the initial box.

Figure 5. The solution is refined using the SIVIA algorithm with separators.
The resulting subpaving after SIVIA algorithm was applied contains three types of boxes:
- $\mathcal{X}_{\text{in}}$: a box which is completely included in the solution set $\mathcal{X}$;
- $\mathcal{X}_{\text{out}}$: a box which does not contain any part of the solution;
- $\mathcal{X}_{\text{boundary}}$: a box that contains regions from both the inside and the outside the solution.

The two bracketing subpavings are defined as:

\[ \mathcal{X} = \mathcal{X}_{\text{in}}, \quad \overline{\mathcal{X}} = \mathcal{X}_{\text{in}} \cup \mathcal{X}_{\text{boundary}}. \]

The interval computations employed to generate the previous figures were performed by using an interval computation library for C++ (Ibex [48] [49]).

2.4 Eulerian methods based on Lyapunov theory. An interval approach.

**Definition 2.1.** The zero solution $\mathbf{x}(t) \equiv 0$ is said to be Lyapunov stable if for all $\varepsilon > 0$ there exists $\delta = \delta(\varepsilon) > 0$ such that if $\|\mathbf{x}(0)\| < \delta$, then $\|\mathbf{x}(t)\| < \varepsilon$ for $t \geq 0$.

**Theorem 2.1 (Lyapunov's Theorem).** Consider a nonlinear dynamical system $\dot{\mathbf{x}} = f(\mathbf{x}(t))$ and assume that there exists a continuously differentiable function $V: \mathcal{D} \rightarrow \mathbb{R}$ such that:

\[
\begin{align*}
V(0) &= 0, \\
V(\mathbf{x}) &\geq 0, \quad \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0, \\
\dot{V}(\mathbf{x})f(\mathbf{x}) &\leq 0, \quad \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0, \\
V(\mathbf{x}) &\to \infty \quad \text{as} \quad \|\mathbf{x}\| \to \infty.
\end{align*}
\]

Then the zero solution of the nonlinear dynamical system is *globally asymptotically stable*.

**Theorem 2.2 (Barbashin-Krasovskii-LaSalle Theorem).** Consider a nonlinear dynamical system $\dot{\mathbf{x}} = f(\mathbf{x}(t))$, assume that $\mathcal{D}_c \subset \mathcal{D}$ is a compact positively invariant set with respect to the dynamical system, and assume there exists a continuously differentiable function $V: \mathcal{D}_c \rightarrow \mathbb{R}$ such that $\dot{V}(\mathbf{x})f(\mathbf{x}) = 0$, $\mathbf{x} \in \mathcal{D}_c$. Let $\mathcal{R} \triangleq \{ \mathbf{x} \in \mathcal{D}_c | \dot{V}(\mathbf{x})f(\mathbf{x}) = 0 \}$ and let $\mathcal{M}$ be the largest invariant set contained in $\mathcal{R}$. If $\mathbf{x}(0) \in \mathcal{D}_c$ then $\mathbf{x}(t) \to \mathcal{M}$ as $t \to \infty$.

The above theorem assumes the existence of a compact invariant set $\mathcal{D}_c \subset \mathcal{D}$ for the nonlinear system. The next theorem [50] does not require this assumption.

**Theorem 2.3.** Consider a nonlinear dynamical system $\dot{\mathbf{x}} = f(\mathbf{x}(t))$ and assume that there exists a continuously differentiable function $V: \mathbb{R}^n \rightarrow \mathbb{R}$ such that:

\[
\begin{align*}
V(0) &= 0, \\
V(\mathbf{x}) &\geq 0, \quad \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0, \\
\dot{V}(\mathbf{x})f(\mathbf{x}) &\leq 0, \quad \mathbf{x} \in \mathbb{R}^n.
\end{align*}
\]
Let, $\mathcal{R} \triangleq \{ \mathbf{x} \in \mathbb{R}^n \mid \dot{V}(\mathbf{x})f(\mathbf{x}) = 0 \}$ and let $\mathcal{M}$ be the largest invariant set contained in $\mathcal{R}$. Then all solutions $\mathbf{x}(t)$, $t \geq 0$, of the nonlinear dynamical system that are bounded approach $\mathcal{M}$ as $t \to \infty$.

The next theorem [50] guarantees the global asymptotic stability of a nonlinear dynamical system.

**Theorem 2.4.** Consider a nonlinear dynamical system $\dot{\mathbf{x}} = f(\mathbf{x}(t))$ and assume that there exists a continuously differentiable function $V: \mathbb{R}^n \to \mathbb{R}$ such that:

- $V(0) = 0$,
- $V(\mathbf{x}) \geq 0$, $\mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0$,
- $\dot{V}(\mathbf{x})f(\mathbf{x}) \leq 0$, $\mathbf{x} \in \mathbb{R}^n$,
- $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$.

Furthermore, assume that the set $\mathcal{R} \triangleq \{ \mathbf{x} \in \mathbb{R}^n \mid \dot{V}(\mathbf{x})f(\mathbf{x}) = 0 \}$ contains no invariant set other than the set $\{0\}$. Then the zero solution $\mathbf{x}(t) \equiv 0$ of the nonlinear dynamical system is globally asymptotically stable.

Theorems 2.2, 2.3 and 2.4 are known as the invariant set theorems. They are different from Lyapunov’s theorem because they relax the strict negative definition of the Lyapunov derivative $\dot{V}(\mathbf{x})$, $\mathbf{x} \in \mathcal{D}$, for both local and global asymptotic stability. It can be seen from the above theorems that one of the key elements in analysing the stability of nonlinear systems is to find a Lyapunov function, which is not always possible in practice.

An intuitive illustration for the above theorems is presented in Figure 6. The figure illustrates a trajectory of a pendulum dynamical system. The trajectory can have an energy level equal to zero multiple times. The state space is $\mathbb{R}^2$, and for this system, the set $\mathcal{R}$ is defined as $\mathcal{R} = \{x_1 \in \mathbb{R}, x_2 = 0 \}$. By using LaSalle theorem, the largest invariant set included in the set $\mathcal{R}$ is the set $\mathcal{M}$ containing the equilibrium point. Hence, the only stable equilibrium point is $(x_1, x_2) = (0,0)$.

![Figure 6. A trajectory starting in (-1.5,-1.5) has $\dot{V}(x_1, x_2) = 0$ every time $x_2 = 0$.](image)
If the system is uncertain, the nonlinear dynamic systems can be described by a differential inclusion $\dot{x} \in F(x)$. New tools should be employed to study the stability for a differential inclusion. The idea of V-Stability is derived from Lyapunov stability theory [50] and influenced by the book of Aubin and Frankowska [51].

**Definition 2.2.** (V-Stability [20]) Consider a differentiable function $V: \mathbb{R}^n \rightarrow \mathbb{R}$. The system $\dot{x} = f(x)$ is said to be V-Stable if there exist $\varepsilon > 0$ such that:

\[
\text{If } V(x) \geq 0 \Rightarrow \dot{V}(x) \leq \varepsilon < 0,
\]

where $\dot{V}(x) = \frac{\partial V}{\partial x}(x)f(x)$.

$V(x)$ appears similar to a Lyapunov function candidate, but, in reality, it is not exactly the same. The concept of V-Stability is illustrated in Figure 7 [20], where the black arrows represent the vector field of the dynamical system. When $V(x) \geq 0$ then all system trajectories must be attracted by the grey region (bubble) limited by the level curve $V(x) = 0$, which is similar to Lyapunov-stability, because $\dot{V}(x) \leq \varepsilon < 0$. The difference arises when the state lies in the grey bubble where $V(x)$ becomes negative. Then, the system trajectory can follow any behaviour, for instance, a limit cycle, as illustrated in the figure. The V-Stability definition states that when the scalar differentiable function $V(x)$ is positive then it will strictly decrease along the level curves. In this case the trajectory will be attracted by the grey region which is an invariant set. Once inside the attraction zone, the trajectory will stay inside forever and can follow any behaviour. Consequently, the notion of V-Stability is weaker than the stability in the sense of Lyapunov [5].

![Figure 7. V-Stability for $V(x) \in [0, \infty]$.](image)

**Consequence of the Lyapunov theorem.** The stability problem of a dynamic system can be represented as a set of inequalities. This is very important because the set of inequalities can be solved...
easily using numerical methods. The following theorem proposes a similar idea, by showing that V-
Stability can be determined from the inconsistency of a set of inequalities.

**Theorem 2.5.** Consider a nonlinear time independent system $\dot{x} = f(x)$. If the set of constraints:

\[
\begin{align*}
\frac{\partial V}{\partial x}(x)f(x) & \geq 0 \\
V(x) & \geq 0
\end{align*}
\]

is inconsistent $\iff \dot{x} = f(x)$ is V-stable.

The Eulerian methods for stability are addressed in this thesis in Chapters 4 and 5 in the
context of differential inclusion. The stability of mechanical systems for which Lyapunov functions
candidates are based on system energy is analysed in Chapter 4. The stability for the differential
inclusion is analysed based on V-Stability theory. V-Stability suffers the same problems as the classical
Lyapunov theory meaning that if the Lyapunov function derivative is only negative semidefinite then
LaSalle theorem is needed to prove the asymptotic stability. However, LaSalle cannot be applied for
dynamic systems with uncertainties and new methods based on perturbation theory and Lyapunov
theory are proposed to solve the LaSalle theorem for systems with uncertainties. Moreover, in
Chapter 3, the Eulerian method is used to tune the parameters for the Sliding Mode Controller and to
find the regions of stability for the control system.

2.5 Lagrangian methods based on interval simulation

When using Lagrangian methods, the exact set of estimated states is difficult or, in the
majority of the cases, impossible to compute. In general, we do not have an explicit formula for the
integration of the differential equation; hence we can only provide an approximate solution based on
some numerical methods. In the literature, two general approaches exist to approximate the solutions
for a differential equation and are based on:

- simple regions to approximate the set of states, and;
- a set of punctual trajectories.

In the first approach, the estimated states are bounded at every iteration by a region which is
easy to represent in a computer (box, ellipsoid, zonotope, sub-pavings). In the literature, there are
several existing algorithms used to propagate the set of states from one step to the next one. This
approach is affected by the wrapping effect and the multiple-incidence problem. As it was shown in
[43] [44] the methods used in this approach are mainly based on propagating, first the centre of the
uncertain region (box, ellipsoid, zonotope etc.) using the mean value theorem, and then the radius of the uncertain set. The main algorithms will be introduced in Section 2.7.

In the second approach, the approximated regions are obtained based on propagating real trajectories. In this way, the method overcomes the wrapping effect. However, the problem of the interval function range evaluation cannot be avoided. Section 2.8 will introduce two algorithms following this approach.

It was proved by Cugueró [16] that for discrete-time interval systems that satisfy the isotonic property, the wrapping effect does not affect the interval propagation. For those types of systems, both approaches will provide the same simulation results. However, when this property is not satisfied, then:

- if the system is contractive, according to Puig [52] the interval simulation will be overestimated, but with a stable wrapping, i.e. the wrapping effect does not increase in time;
- if the system is not contractive, according to [43], the interval simulation will be affected by an unstable wrapping effect.

2.6 Problems in interval simulations

2.6.1 Range evaluation of an interval function

When predicting the behaviour of a dynamic system in discrete time, at any time instant, the state function is evaluated by computing the range of an interval function. The simplest way to evaluate the range of an interval function is by direct application of the interval arithmetic as it was proposed by Moore in its seminal work published in 1966 [53]. An overestimation occurs when multiple instances of the same variable appear in the same expression.

Example: If \( x \in [-1,1] \), then the evaluation of \( z = x - x \) using interval arithmetic should be 0, but the result is \([-2,2]\). This overestimation phenomena was first observed by Moore in 1966 [53] and it is known as the multi-incidence problem. One of the following methods are proposed in the literature to overcome this problem:

- Combining the interval arithmetic methods with a branch and bound algorithm [54];
- Evaluating the range of an interval function by solving two optimisation problems (a minimisation and a maximisation) using numerical methods. But classical numerical
optimisation algorithms can only guarantee local optimums since they are gradient based. Global optimums can only be obtained if the optimisation problems associated with the range evaluation are convex [55];

- Employing global optimisation methods based on branch and bound [41] to guarantee global optimums for nonlinear and non-convex optimisation problems;
- Affine arithmetic is a good tool which can be used to overcome the multi-incidence problem for linear systems.

### 2.6.2 Wrapping effect

The problem of wrapping is related to the use of a crude approximation, such as a box, for the uncertainty region of the states and then using one-step propagation. At every iteration, the solution set is wrapped into a superset feasible to be constructed and represented in a computer (box, zonotope, ellipsoid, etc.). The overestimation of the wrapped set is proportional to its radius. Moreover, due to the duality approximation and then iteration of the box, the resulting wrapping effect is illustrated in Figure 8 [14]. The wrapping effect is completely unrelated to the stability properties of the dynamic system. Stable systems approximations are shown to exhibit exponentially fast-growing enclosures, thus becoming useless for practical purposes.

![Figure 8. Wrapping effect](image)

The wrapping effect and its application to the validated solution of initial value problems were first observed in the early 60s [53]. Since then, several approaches have been proposed to reduce the effect of the wrapping in Lagrangian methods:

- rotating the state space of the interval system as it was proposed first by Moore [53] using a change of coordinates and then, later, proposed by Lohner [32] using a QR-factorisation.
• approximating the state space region using better shapes than boxes, for example, using ellipsoids as it was proposed by Neumaier [56] or using zonotopes, as it was proposed by Kühn [14].

However, as it is shown by Stancu [44], all these approaches do not succeed in overcoming the wrapping effect except in some particular cases. This is a common problem for all Lagrangian methods proposed in the literature.

2.7 Approaches using regions

All algorithms described in this section produce only an outer (conservative) solution for interval observation. The propagation mechanism used in these algorithms produces an approximated region that includes all possible states based on previous approximated regions. However, spurious states are also included when approximating the solution. Therefore, the introduction of spurious states will inflate the uncertainty region (wrapping effect), resulting in a superset of solutions that provides an outer approximation. In some situations, this leads to an unstable Lagrangian method for integration.

The main algorithms that propagate regions, proposed by Moore(1966) [53], Lohner(1987) [32], Neumaier(1993) [56], Kühn(1998) [14], and Sandretto (2016) [15], are presented and compared in [43] and briefly introduced in this section.

Moore’s algorithm

The first Lagrangian method based on region propagation was proposed by Moore and it is based on the interval mean-value theorem. To avoid the wrapping effect, Moore has proposed, at every iteration, a moving coordinate system that matches the solution set.

However, this method still suffers from the wrapping effect for non-isotonic and non-contractive systems and for some ill-conditioned systems, as for example, systems with eigenvalues with very different magnitudes [34].
Lohner’s algorithm

This algorithm is similar to Moore’s algorithm but it introduced a QR factorisation for ill-conditioned systems when Moore’s algorithm is highly affected by the wrapping effect. In this case, Moore’s algorithm is modified as it was proposed by Lohner in 1987 [32].

The moving orthogonal coordinate system at every iteration better matches with the solution set reducing in this way the wrapping effect comparing with the initial Moore’s algorithm.

Neumaier’s algorithm

Instead of using a box to enclose the uncertainties for a dynamical system Neumaier in 1993 [56] has proposed to use ellipsoids.

This algorithm is a relative algorithm, as Moore and Lohner algorithms, because it consists in separately propagating the centre and the radius of the ellipsoid. One of the advantages of this algorithm is that, in some cases, it is more efficient to approximate the uncertainty region with ellipsoids instead of the crude approximation using boxes. Moreover, the rotation of the ellipsoid in the state space is implicit and it is not necessary to make additional computations as in the case of Lohner’s algorithm.

Kühn’s algorithm

Kühn’s algorithm is based on approximating the uncertain region of system states using zonotopes.

Because of the complex shape of a zonotope, Kühn’s algorithm can manage the uncertainty propagation better than Moore’s, Lohner’s and Neumaier’s algorithms. However, if the system is non isotone and non-contractive, the zonotope propagation is unstable as it was shown in [43] [44].

Guaranteed integration

This is the most recent algorithm which belongs to the Lagrangian methods for computing the solution for a differential inclusion [15]. The guaranteed integration algorithm is based on Picard and Brouwer theorems [21].
2.8 Approaches using real trajectories

Another important method for solving differential equations with uncertainty is based on propagating real trajectories instead of regions. Approaches that are based on propagating real trajectories are not affected by the wrapping effect. Two algorithms are mainly used in the literature for trajectory propagation. The first algorithm provides an inner approximation for the solution of the differential inclusion and it was proposed by Kolev (1993) [46]. The algorithm consists in propagating only the vertices of the uncertain region. The second algorithm was developed by Stancu [57] and it was proposed in the context of the worst-case simulation by solving an optimisation problem. Stancu’s algorithm provides an outer approximation for the solution of the differential inclusion.

2.9 Conclusions

Lagrangian and Eulerian methods, to deal with stability analysis as well as for approximating the solutions of differential equations and differential inclusions were presented in this chapter. Several algorithms belonging to these categories (region based algorithms and trajectory based algorithms) were discussed. Region propagation algorithms are based on propagating the uncertainty region for system states using one-step-ahead recursion. The main problem of region-based methods is the wrapping effect. This problem affects all algorithms based on region propagation except some special cases, i.e. when the system satisfies the isotonic property or is contractive. In some particular cases, advanced algorithms such as Lohner’s, Neumaier’s, Kühn’s, and guaranteed integration algorithms can reduce the effect of wrapping. However, these algorithms fail when the system does not fulfil the contractivity, or require a large number of bisections to reduce the wrapping effect as in the case of guaranteed integration. In the case of algorithms based on trajectory propagation instead of region propagation, an inner solution can be obtained using Kolev’s algorithm. The computation time is low in this case, however the result is not guaranteed since the algorithm provides only an inner approximation for the solution. The outer approximation proposed by Stancu is a guaranteed solution, but it is expensive from the computational point of view since a nonlinear optimisation problem must be solved at every sampling time.
Chapter 3: Sliding mode control of a differential drive robot

3.1 Introduction

In the last years, the usage of robots in difficult environments is increasing as they can replace humans in performing dangerous tasks. Underwater robots [58] are being used lately for tasks like seabed exploration or performing repairs on underwater oil rigs. Some robots are now used in completely remote and unknown environments like in the case of Mars rover which has to work in harsh environments, without human intervention and for a long time. If the robots are operating in a dangerous environment, they are also more prone to failures and in this case, the physical system and the control law have to be designed taking this aspect into consideration. Furthermore, because the robots can be operated in environments which are remote or inaccessible to humans, faults can lead to the loss of the robot. Most of the times, the robots are expensive and sometimes even unique which makes their recovery from the working environment paramount.

One particular field where robots have been used for a long time is the nuclear environment. The risks involved for humans are too high in this case and robots (or manipulators) [59] [60] are being used for some operations almost since the beginning of the nuclear era.

In our days, robots are required to perform more and more difficult and sophisticated tasks in the nuclear environment. The nuclear radiation can damage electronic circuits [61] which means the robot is prone to electronic failures and this has to be considered when designing the electronics of the robot [62], but also the control algorithm. One interesting aspect, in this case, is that the electronics can heal after being removed from an area with high radiation, which makes fault tolerant control even more important in this situation.

3.1.1 Sliding mode fault tolerant control for mobile robots

The usage of wheeled mobile robots in hazardous environments or for long-time operations requires an increase of their robustness and tolerance against faults by employing Fault Tolerant Control (FTC) mechanisms. FTC aims to maintain current performances close to desirable ones and preserve stability conditions in the presence of faults ([63], [64]). The existing FTC design techniques can be classified, depending on how they deal with faults, in two categories: passive and active [65]. The passive FTC techniques take the fault into account as a system disturbance during the design
phase. Thus, the control law has inherent fault-tolerant capabilities within certain margins, allowing the system to cope with the fault occurrence [66]. The active FTC techniques can compensate the faults either by selecting a pre-calculated control law or by synthesising a new control strategy online. Successful multiple-model-based FTC strategies are used in [67], learning-based in [68] and adaptive backstepping-based in [65], among others. Fault hiding is another appealing strategy of active fault tolerant control which considers an intermediate level between the system and the controller that hides the fault and avoids readjusting the controller.

FTC has only been recently applied to mobile robots. A few references can be found in the literature and are summarised in [69]. The FTC problem, used in robotics to increase their robustness against possible faults, has been addressed in [70] [71]. For instance, four-wheeled holonomic mobile robots with four active wheels have the characteristic that they can still operate with three active wheels in case some malfunctioning in one active wheel has been detected [72]. This makes them good setups for testing techniques that provide fault tolerance against actuator faults. Just recently, FDI and FTC approaches have been considered for multi-robot systems in [73], where a distributed controller-observer architecture was proposed. This allows each robot to estimate the global system state using local communication.

3.1.2 Research objectives and contributions

SMC is a known robust control technique based on the model of the controlled system. In the following sections, SMC will be used to study and develop robust control structures for a two-wheel differential drive robot.

For the real scenarios, a robotic platform built in the Autonomous Systems Lab at The University of Manchester is used as a testbed for the fault tolerant control algorithms. This robotic platform was developed in-house because it has some advantages when compared with off the shelf solutions: it allows easy access to all components as well as to all available measurements, it allows artificial fault generation for testing new control algorithms, it is easy to configure and add new modules or sensors.
The performance of the control structures is first tested using simulations. A more complex dynamic model is developed in the second section of this chapter for this purpose.

Developing a simulation for the Wheeled Mobile Robot (WMR) is required for several reasons:

- Due to space and safety restrictions, the real robotic platform can only be used in specific lab environments and for a limited amount of time;
- Fault generation is not easy to achieve on the real platform. Some of the fault scenarios can only be tested in simulation.

The SMC structures are addressed in simulation, and new torque control strategy is proposed to solve the encountered problems. All new control methods are then tested on the real robotic platform to validate the results. Based on the knowledge gathered while analysing various SMC approaches, a new fault hiding approach is introduced in Section 3.4.

In Section 3.5, a novel approach, based on interval methods, to study the stability of the SMC when dealing with uncertainty is proposed.
3.2. Dynamic model of a two-wheeled mobile robot

3.2.1 Coordinate systems

There have been many approaches in the past for dealing with the trajectory tracking problem [74], but using only the kinematic model or simpler dynamic models. Although the kinematic approach has proved its reliability, it will not suffice for the analysis of robust controllers dealing with faults. The Euler-Lagrange approach offers one of the most complete and versatile solutions for simulating dynamical systems, in our case, a non-holonomic two-wheeled differential drive mobile robot.

The first step for defining the dynamic model is to describe the coordinate systems used for the robot motion and the set of constraints derived from the non-holonomic requirement of the robot.

In our case we have two coordinate systems:

- the world frame coordinate system which is fixed and used as a reference frame (it does not move with the robot);
- the robot frame with the origin in the middle of the robot wheel axis. This is a local frame attached to the robot and moves with it.

The robot pose can now be defined in both coordinate systems: world frame \( \mathbf{q}^w = \begin{bmatrix} x_w \\ y_w \\ \theta_w \end{bmatrix} \) and robot frame \( \mathbf{q}^r = \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix} \).

The position of any point on the robot can also be defined in the robot frame or world frame as follows:

- world frame \( \mathbf{X}^w = \begin{bmatrix} x^w \\ y^w \end{bmatrix} \) and robot frame \( \mathbf{X}^r = \begin{bmatrix} x^r \\ y^r \end{bmatrix} \).

This position can be converted from one frame to another using a transformation matrix (orthogonal rotation matrix):

\[
\mathbf{X}^w = \mathbf{R}(\theta)\mathbf{X}^r, \quad \text{where} \quad \mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

This transformation is also used for converting velocities from one frame to another, which is useful and will be used in the next section:

\[
\dot{\mathbf{X}}^w = \mathbf{R}(\theta)\dot{\mathbf{X}}^r
\]
3.2.2 Kinematic constraints

The motion of the non-holonomic mobile robot is governed by the two constraints defined below.

The first constraint imposes no lateral motion for the robot on its trajectory. This means that the robot can only have linear velocity (forward/backward motion) and angular velocity (right/left rotation). The constraint can be described in the robot frame as follows:

\[ \dot{y}_r = 0, \]

Which, in other words, means that the lateral velocity in robot frame must be zero. This can be converted to world frame by using the transformation matrix which gives:

\[-\dot{x}_w \sin(\theta) + \dot{y}_w \cos(\theta) = 0\] (3.2)

The second constraint imposes no wheel rolling slip. This means that each wheel should have an angular velocity proportional with the robot velocities (no longitudinal slip occurs at the contact point of the wheel with the ground). The linear velocities of the wheels can be expressed in the world frame as follows:

\[
\begin{align*}
    v_{pR} &= R_R \dot{\phi}_R \\
    v_{pL} &= R_L \dot{\phi}_L
\end{align*}
\] (3.3)

These velocities can also be expressed in the world frame using robot velocities:

\[
\begin{align*}
    \dot{x}_{pR} &= \dot{x}_w + L \dot{\theta} \cos \theta \\
    \dot{y}_{pR} &= \dot{y}_w + L \dot{\theta} \sin \theta \\
    \dot{x}_{pL} &= \dot{x}_w - L \dot{\theta} \cos \theta \\
    \dot{y}_{pL} &= \dot{y}_w - L \dot{\theta} \sin \theta
\end{align*}
\] (3.4)

And the velocities of the contact points in world frame for each wheel:

\[
\begin{align*}
    v_{pR} &= \dot{x}_{pR} \cos \theta + \dot{y}_{pR} \sin \theta \\
    v_{pL} &= \dot{x}_{pL} \cos \theta + \dot{y}_{pL} \sin \theta
\end{align*}
\] (3.5)

From the above equations, we can derive three constraint equations:

\[
\begin{align*}
    -\dot{x}_w \sin \theta + \dot{y}_w \cos \theta &= 0 \\
    \dot{x}_w \cos \theta + \dot{y}_w \sin \theta + L \dot{\theta} - R_R \dot{\phi}_R &= 0 \\
    \dot{x}_w \cos \theta + \dot{y}_w \sin \theta - L \dot{\theta} - R_L \dot{\phi}_L &= 0
\end{align*}
\] (3.6)

And in the matrix form:

\[ \Lambda(q) \dot{q} = 0 \]

where \( q \) is the general coordinate vector and
\[
\dot{q} = \begin{bmatrix}
\dot{x}_w \\
\dot{y}_w \\
\dot{\theta} \\
\dot{\phi}_R \\
\dot{\phi}_L
\end{bmatrix}; \quad \Lambda(q) = \begin{bmatrix}
-sin\theta & cos\theta & 0 & 0 & 0 \\
-cos\theta & sin\theta & L & -R_R & 0 \\
-cos\theta & sin\theta & -L & 0 & -R_L
\end{bmatrix}
\]  

(3.7)

Figure 10. Diagram of the WMR

3.2.3 The WMR dynamic model using the Euler-Lagrange approach

The dynamical model of a non-holonomic differential drive mobile robot can be described by the following equation:

\[
M(q) \ddot{q} + V(q, \dot{q}) \dot{q} + \tau_d = B(q) \tau - F_R(\dot{q}) - \Lambda^T(q) \lambda
\]  

(3.8)

where:

- \( M(q) \) – \( nxn \) inertia matrix;
- \( V(q, \dot{q}) \) – coriolis and centripetal matrix;
- \( F_R(\dot{q}) \) – surface and rolling friction matrix;
- \( \tau_d \) – unknown disturbances and unstructured unmodeled dynamics;
- \( B(q) \) – input matrix;
- \( \tau \) – input vector;
- \( \Lambda^T \) – kinematic constraints matrix;
- \( \lambda \) – lagrange multiplier;

Lagrange dynamical modelling can be used for deriving the equations of motion for any mechanical system. This method was first used by Lagrange and it uses the kinetic and potential energies to express dynamics of a system.
\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_i} \right) + \frac{\partial L}{\partial q_i} = F - \Lambda^T(q) \lambda
\] (3.9)

where \( L = T - V \) is the Lagrangian function, \( T \) is the kinetic energy, \( V \) is the potential energy of the system and \( q_i \) is the generalized coordinates vector.

In order to derive the dynamic model of the robot, we have to define the kinetic and potential energy for each component we need to model. Because the robot moves only in the \((X, Y)\) plane the potential energy can be considered zero. The kinetic energy of the robot is defined in our case by the sum of the kinetic energy of the robot body and the kinetic energy of the wheels together with the actuators.

The kinetic energy of the robot body:
\[
T_B = \frac{1}{2} M_B V_B^2 + \frac{1}{2} I_B \dot{\theta}^2
\] (3.10)

The kinetic energy of the right and left wheel:
\[
T_{WR} = \frac{1}{2} M_W V_{WR}^2 + \frac{1}{2} I_M \dot{\theta}^2 + \frac{1}{2} I_W \dot{\phi}_R^2
\]
\[
T_{WL} = \frac{1}{2} M_W V_{WL}^2 + \frac{1}{2} I_M \dot{\theta}^2 + \frac{1}{2} I_W \dot{\phi}_L^2
\]

where \( M_B \) is the mass of the robot body without the wheels and actuators, \( M_W \) is the mass of each driving wheel including the actuator, \( I_B \) is the moment of inertia of the robot body relative to the center of mass, \( I_M \) the moment of inertia of each wheel relative to the rotation of the robot, \( I_W \) is the moment of inertia of each wheel relative to its axis (including the inertia of the actuator).

In order to derive the velocity components required for the kinetic energy we can write the position of the center of mass of the robot as follows:
\[
\begin{align*}
    x_B &= x_w + d \cos \theta \\
    y_B &= y_w + d \sin \theta
\end{align*}
\] (3.12)

\[
V_B^2 = \dot{x}_B^2 + \dot{y}_B^2 = \dot{x}_w^2 + \dot{y}_w^2 - 2 \dot{x}_w d \dot{\theta} \sin \theta + 2 \dot{y}_w d \dot{\theta} \cos \theta + d^2 \dot{\theta}^2
\] (3.13)

The velocities of the right and left wheel in world frame coordinates are derived as follows:
\[
\begin{align*}
    \dot{x}_{pR} &= \dot{x}_w + L \dot{\theta} \cos \theta \\
    \dot{y}_{pR} &= \dot{y}_w + L \dot{\theta} \sin \theta \\
    \dot{x}_{pL} &= \dot{x}_w - L \dot{\theta} \cos \theta \\
    \dot{y}_{pL} &= \dot{y}_w - L \dot{\theta} \sin \theta
\end{align*}
\] (3.14)
The linear velocity for each wheel in world frame coordinates is:
\[
\begin{align*}
V_{WR}^2 &= \dot{x}_{pR}^2 + \dot{y}_{pR}^2 = \dot{x}_W^2 + \dot{y}_W^2 + 2\dot{x}_w L \dot{\theta} \cos \theta + 2\dot{y}_w L \dot{\theta} \sin \theta + L^2 \dot{\theta}^2 \\
V_{WL}^2 &= \dot{x}_{pL}^2 + \dot{y}_{pL}^2 = \dot{x}_W^2 + \dot{y}_W^2 - 2\dot{x}_w L \dot{\theta} \cos \theta - 2\dot{y}_w L \dot{\theta} \sin \theta + L^2 \dot{\theta}^2
\end{align*}
\] (3.15)

The total kinetic energy of the robot becomes:
\[
T = \frac{1}{2} M (\dot{x}_W^2 + \dot{y}_W^2) - M_B d \dot{\theta} (\dot{y}_w \cos \theta - \dot{x}_w \sin \theta) + \frac{1}{2} I_W (\dot{\phi}_R^2 + \dot{\phi}_L^2) + \frac{1}{2} I \dot{\theta}^2
\] (3.16)

where: \( M = M_B + 2M_W \) and \( I = I_B + M_B d^2 + 2M_W L^2 + 2I_M \)

We can now apply the Lagrange function to the energy equation above, for \( L = T \):
\[
\begin{align*}
M\ddot{x}_w - M_B d \dot{\theta} \sin \theta - M_B d \dot{\theta}^2 \cos \theta &= C_1 \\
M\ddot{y}_w - M_B d \dot{\theta} \cos \theta - M_B d \dot{\theta}^2 \sin \theta &= C_2 \\
I \ddot{\theta} - M_B d \dot{x}_w \sin \theta + M_B d \dot{y}_w \cos \theta &= C_3 \\
I_W \ddot{\phi}_R &= \tau_R + C_4
\end{align*}
\] (3.17)

where \((C_1, C_2, C_3, C_4, C_5)\), are coefficients related to the kinematic constraints, which can be written in terms of the Lagrange multipliers vector \( \lambda \) and the kinematic constraints matrix \( \Lambda(q) \).

The dynamic model can be written in the general matrix form:
\[
M(q) \ddot{q} + V(q, \dot{q}) \dot{q} = B(q) \tau - F_R(q) - \Lambda^T(q) \lambda
\] (3.18)

where:
\[
M(q) = \begin{bmatrix}
M & 0 & -M_B d \sin \theta & 0 & 0 \\
0 & M & M_B d \cos \theta & 0 & 0 \\
-M_B d \sin \theta & M_B d \cos \theta & I & 0 & 0 \\
0 & 0 & 0 & I_W & 0 \\
0 & 0 & 0 & 0 & I_W
\end{bmatrix};
\]
\[
V(q, \dot{q}) = \begin{bmatrix}
0 & 0 & -M_B d \dot{\theta} \cos \theta & 0 & 0 \\
0 & 0 & -M_B d \dot{\theta} \sin \theta & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix};
\]
\[ B(q) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Lambda^T(q)\lambda = \begin{bmatrix} -\sin\theta & \cos\theta & \cos\theta \\ \cos\theta & \sin\theta & \sin\theta \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}; \quad F_R(q) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]

Equation (3.18) can be written in a different way which is easier to use in simulation. This is done by replacing the general coordinate vector with a reduced version that contains only the angular velocities of the wheels:

\[ \eta = \begin{bmatrix} \dot{\phi}_R \\ \dot{\phi}_L \end{bmatrix} \]

The generalized coordinates vector \( q \) can be expressed as a function of \( \dot{\eta} \) by using a transformation matrix:

\[ \begin{bmatrix} \dot{x}_w \\ \dot{y}_w \\ \dot{\theta} \\ \dot{\phi}_R \\ \dot{\phi}_L \end{bmatrix} = \begin{bmatrix} \frac{R_R \cos\theta}{2} & \frac{R_L \cos\theta}{2} \\ \frac{R_R \sin\theta}{2} & \frac{R_L \sin\theta}{2} \\ \frac{2L}{R_R} & \frac{2L}{R_L} \\ \frac{1}{R_R} & \frac{0}{R_L} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi}_R \\ \dot{\phi}_L \end{bmatrix} \]  

This can also be expressed in matrix form:

\[ \ddot{q} = S(q)\dot{\eta} \]  

(3.20)

It can also be verified that the transformation matrix satisfies the following equation:

\[ S^T(q)\Lambda^T(q) = 0 \]  

(3.21)

By taking the time derivative of \( \ddot{q} = S(q)\dot{\eta} \) we get the acceleration vector:

\[ \dddot{q} = \dddot{S}(q)\dot{\eta} + \dddot{S}(q)\eta \]  

(3.22)

We can now replace the generalized coordinate vector \( q \) in equation (3.18).

\[ M(q)\dddot{S}(q)\eta + V(q, \dot{q})[S(q)\eta] = B(q)\tau - F_R(q) - \Lambda^T(q)\lambda \]  

(3.23)

We multiply both sides of the equation (3.23) by \( S^T(q) \) and by rearranging the terms we get the following equation:

\[ S^T(q)M(q)\dddot{S}(q)\eta + S^T(q)\left[M(q)\dddot{S}(q) + V(q, \dot{q})S(q)\right]\eta = S^T(q)B(q)\tau - S^T(q)F_R(q) \]
The equation in $\eta$ can be written as:

$$\ddot{M}(q)\dot{\eta} + \ddot{V}(q, \dot{q})\eta = \ddot{B}(q)\tau - \ddot{F}_R(\dot{q})$$  \hspace{1cm} (3.24)

where the following notations are considered:

$$\ddot{M}(q) = S^T(q)M(q)S(q)$$  \hspace{1cm} (3.25)

$$\ddot{V}(q, \dot{q}) = S^T(q)M(q)\dot{S}(q) + S^T(q)V(q, \dot{q})\dot{S}(q)$$

$$\ddot{B}(q) = S^T(q)B(q)$$

$$\ddot{F}_R(\dot{q}) = S^T(q)F_R(\dot{q})$$

Using the notations (3.25), one can now get the final matrix form:

$$\ddot{M}(q) = \begin{bmatrix}
\frac{R_R \cos \theta}{2} & \frac{R_R \sin \theta}{2} & \frac{R_R}{2L} & 1 & 0 \\
\frac{R_L \cos \theta}{2} & \frac{R_L \sin \theta}{2} & \frac{R_L}{2L} & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
M & 0 & -M_B d \sin \theta & 0 & 0 \\
0 & M & M_B d \cos \theta & 0 & 0 \\
-\frac{M_B d \sin \theta}{2L} & M_B d \cos \theta & I & 0 & 0 \\
0 & 0 & 0 & I_w & 0 \\
0 & 0 & 0 & 0 & I_w \\
\end{bmatrix}
\times S(q)
$$

$$= \begin{bmatrix}
\frac{MR_R \cos \theta}{2} - \frac{M_B dR_R \sin \theta}{2L} & \frac{MR_R \sin \theta}{2L} + \frac{M_B dR_R \cos \theta}{2L} & \frac{IR_R}{2L} & 0 & I_w \\
\frac{MR_L \cos \theta}{2} + \frac{M_B dR_L \sin \theta}{2L} & \frac{MR_L \sin \theta}{2L} - \frac{M_B dR_L \cos \theta}{2L} & \frac{IR_L}{2L} & 0 & I_w \\
\end{bmatrix}
\begin{bmatrix}
\frac{R_R \cos \theta}{2} & \frac{R_R \sin \theta}{2} \\
\frac{R_L \cos \theta}{2} & \frac{R_L \sin \theta}{2} \\
\end{bmatrix}
\times \begin{bmatrix}
\frac{2}{R_R} & \frac{2}{R_L} \\
\frac{2}{R_R} & \frac{2}{R_L} \\
\end{bmatrix}
\begin{bmatrix}
\frac{2}{R_R} \cos \theta & \frac{2}{R_L} \cos \theta \\
\frac{2}{R_R} \sin \theta & \frac{2}{R_L} \sin \theta \\
\end{bmatrix}
$$

$$\ddot{M}_{11} = \frac{MR_R^2 \cos^2 \theta}{4} - \frac{M_B dR_R^2 \sin \theta \cos \theta}{4L} + \frac{MR_R^2 \sin^2 \theta}{4L} + \frac{M_B dR_R^2 \sin \theta \cos \theta}{4L} + \frac{IR_R^2}{4L^2} + I_w = \frac{R_R^2}{4} \left(M + \frac{I}{L^2}\right) + I_w$$

$$\ddot{M}_{12} = \frac{MR_L R_R \cos^2 \theta}{4} - \frac{M_B dR_R R_L \sin \theta \cos \theta}{4L} + \frac{MR_L R_L \sin^2 \theta}{4L} + \frac{M_B dR_R R_L \sin \theta \cos \theta}{4L} - \frac{IR_R R_L}{4L^2}$$

$$\ddot{M}_{21} = \frac{MR_L R_L \cos^2 \theta}{4} + \frac{M_B dR_R R_L \sin \theta \cos \theta}{4L} + \frac{MR_L R_L \sin^2 \theta}{4L} - \frac{M_B dR_R R_L \sin \theta \cos \theta}{4L} - \frac{IR_R R_L}{4L^2}$$

$$\ddot{M}_{22} = \frac{MR_L^2 \cos^2 \theta}{4} + \frac{M_B dR_L^2 \sin \theta \cos \theta}{4L} + \frac{MR_L^2 \sin^2 \theta}{4L} - \frac{M_B dR_L^2 \sin \theta \cos \theta}{4L} + \frac{IR_L^2}{4L^2} + I_w = \frac{R_L^2}{4} \left(M + \frac{I}{L^2}\right) + I_w$$

$$\ddot{M}(q) = \begin{bmatrix}
\frac{R_R^2}{4} \left(M + \frac{I}{L^2}\right) + I_w \\
\frac{R_R R_L}{4} \left(M - \frac{I}{L^2}\right) \\
\frac{R_L^2}{4} \left(M + \frac{I}{L^2}\right) + I_w \\
\frac{R_R R_L}{4} \left(M - \frac{I}{L^2}\right) \\
\end{bmatrix}$$

53
\[
\dot{S}(q) = \begin{bmatrix}
\frac{R_R \dot{\theta} \sin \theta}{2} & -\frac{R_L \dot{\theta} \sin \theta}{2} \\
\frac{R_R \dot{\theta} \cos \theta}{2} & \frac{R_L \dot{\theta} \cos \theta}{2} \\
0 & 0 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\]

\[
S^T(q)M(q)\dot{S}(q) =
\begin{bmatrix}
\frac{R_R \cos \theta}{2} & \frac{R_R \sin \theta}{2} & \frac{R_R}{2L} & 1 & 0 \\
\frac{R_L \cos \theta}{2} & \frac{R_L \sin \theta}{2} & \frac{R_L}{2L} & 0 & 1 \\
0 & 0 & -M_B d \sin \theta & 0 & 0 \\
0 & 0 & M_B d \cos \theta & 0 & 0 \\
0 & 0 & 0 & I_w & 0
\end{bmatrix}
\begin{bmatrix}
\frac{M_B d R_R^2 \dot{\theta}}{4L} \\
-\frac{M_B d R_R R_L \dot{\theta}}{4L} \\
\frac{M_B d R_L^2 \dot{\theta}}{4L}
\end{bmatrix}
\]

\[
S^T(q)V(q, \dot{q})S(q) =
\begin{bmatrix}
\frac{R_R \cos \theta}{2} & \frac{R_R \sin \theta}{2} & \frac{R_R}{2L} & 1 & 0 \\
\frac{R_L \cos \theta}{2} & \frac{R_L \sin \theta}{2} & \frac{R_L}{2L} & 0 & 1 \\
0 & 0 & -M_B d \dot{\theta} \cos \theta & 0 & 0 \\
0 & 0 & -M_B d \dot{\theta} \sin \theta & 0 & 0 \\
0 & 0 & 0 & I_w & 0
\end{bmatrix}
\begin{bmatrix}
0 & 0 & -M_B d \dot{\theta} \cos \theta & 0 & 0 \\
0 & 0 & -M_B d \dot{\theta} \sin \theta & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\times S(q)
\]

\[
V(q, \dot{q}) = S^T(q)M(q)\dot{S}(q) + S^T(q)V(q, \dot{q})S(q) =
\begin{bmatrix}
0 & -M_B d R_R R_L \dot{\theta} \\
-\frac{M_B d R_R R_L \dot{\theta}}{2L} & 0
\end{bmatrix}
\]

54
\[
\bar{B}(q) = \begin{bmatrix}
\frac{R_R \cos \theta}{2} & \frac{R_R \sin \theta}{2} & \frac{R_R}{2L} & 1 & 0 \\
\frac{R_L \cos \theta}{2} & \frac{R_L \sin \theta}{2} & -\frac{R_L}{2L} & 0 & 1 \\
\end{bmatrix} \times \begin{bmatrix}
0 & 0 \\
0 & 0 \\
1 & 0 \\
1 & 1 \\
\end{bmatrix} = \begin{bmatrix} 1 & 0 \\
\end{bmatrix}
\]

\[
\bar{F}_R(q) = \begin{bmatrix}
\frac{R_R \cos \theta}{2} & \frac{R_R \sin \theta}{2} & \frac{R_R}{2L} & 1 & 0 \\
\frac{R_L \cos \theta}{2} & \frac{R_L \sin \theta}{2} & -\frac{R_L}{2L} & 0 & 1 \\
\end{bmatrix} \times \begin{bmatrix}
0 & 0 \\
0 & 0 \\
C_S \text{sign}(\dot{\phi}_R) + C_V \dot{\phi}_R \\
C_S \text{sign}(\dot{\phi}_L) + C_V \dot{\phi}_L \\
\end{bmatrix} = \begin{bmatrix} C_S \text{sign}(\dot{\phi}_R) + C_V \dot{\phi}_R \\
C_S \text{sign}(\dot{\phi}_L) + C_V \dot{\phi}_L \\
\end{bmatrix}
\]

Next, the DC motor model will be used to simulate the actuators and the input torque for the above WMR dynamic model.

\[
J \ddot{\theta} + b \omega = K_{te} l \tag{3.26}
\]

\[
L_e \frac{dl}{dt} + R_e l = V - K_{te} \omega
\]

where:

- \( J \) - rotor moment of inertia;
- \( b \) - viscous friction;
- \( K_{te} \) - the torque and back emf constant;
- \( L_e \) - armature inductance;
- \( R_e \) - armature resistance;
- \( V \) - input voltage;

In our case, we can use a simplified version of this model because we already included the rotor moment of inertia and friction in the WMR model for each wheel. Hence the output torque for each actuator becomes:

\[
\begin{align*}
I_R &= \frac{V_R - N_G K \dot{\phi}_R - R_e I_R}{L_e} ; & \tau_R &= N_G \mu K_{te} I_R \\
I_L &= \frac{V_L - N_G K \dot{\phi}_L - R_e I_L}{L_e} ; & \tau_L &= N_G \mu K_{te} I_L
\end{align*} \tag{3.27}
\]

where:

- \( N_G \) - gearbox multiplication;
- \( \mu \) - gearbox efficiency;

The DC motors are mechanically coupled to the robot wheels through a gearbox. In this case, the actuator motion is proportional to the wheel motion:
\[ \omega_R = N_G \dot{\phi}_R \]
\[ \omega_L = N_G \dot{\phi}_L \] 

3.2.4 Simulation and validation results

The dynamic model was simulated in Matlab using the following physical parameters (this data was measured on the real robot):

Table 3.1

<table>
<thead>
<tr>
<th>Robot parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_R = 0.1145 \text{ m} )</td>
<td>right wheel radius</td>
</tr>
<tr>
<td>( R_L = 0.1145 \text{ m} )</td>
<td>left wheel radius</td>
</tr>
<tr>
<td>( L = 0.245 \text{ m} )</td>
<td>half of wheelbase</td>
</tr>
<tr>
<td>( d = 0.1 \text{ m} )</td>
<td>distance between center of mass and center of rotation</td>
</tr>
<tr>
<td>( M_B = 35 \text{ Kg} )</td>
<td>robot body mass</td>
</tr>
<tr>
<td>( M_W = 0.35 \text{ Kg} )</td>
<td>wheel mass</td>
</tr>
<tr>
<td>( I_B = 1.5 Kg \cdot m^2 )</td>
<td>robot body moment of inertia</td>
</tr>
<tr>
<td>( I_M = 0.001 Kg \cdot m^2 )</td>
<td>wheel moment of inertia around robot axis of rotation</td>
</tr>
<tr>
<td>( I_W = 0.0012 Kg \cdot m^2 )</td>
<td>wheel moment of inertia around actuator axis of rotation</td>
</tr>
<tr>
<td>( C_{SR} = 2.35 )</td>
<td>static resistive torque for the right wheel</td>
</tr>
<tr>
<td>( C_{SL} = 2.73 )</td>
<td>static resistive torque for the left wheel</td>
</tr>
<tr>
<td>( C_{VR} = 0.17 )</td>
<td>viscous resistive torque for the right wheel</td>
</tr>
<tr>
<td>( C_{VL} = 0.25 )</td>
<td>viscous resistive torque for the left wheel</td>
</tr>
</tbody>
</table>

Table 3.2

<table>
<thead>
<tr>
<th>DC motor parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( K_{te} = 0.03 )</td>
<td>torque and back emf constant</td>
</tr>
<tr>
<td>( R_c = 1.8 \Omega )</td>
<td>armature resistance;</td>
</tr>
<tr>
<td>( L_c = 0.08 \text{ H} )</td>
<td>armature inductance</td>
</tr>
<tr>
<td>( N_G = 40 )</td>
<td>gearbox multiplication</td>
</tr>
<tr>
<td>( \mu = 73/100 )</td>
<td>gearbox efficiency</td>
</tr>
</tbody>
</table>

Table 3.3

<table>
<thead>
<tr>
<th>Input voltage</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_R = 9.5 \text{ V} )</td>
<td>Right motor input voltage</td>
</tr>
<tr>
<td>( V_L = 8.5 \text{ V} )</td>
<td>Left motor input voltage</td>
</tr>
</tbody>
</table>
Because there is no known system identification technique that can be directly used with this robotic system, the parameters were determined by direct measurement (mass, size, armature resistance), indirect measurement (friction, torque constant etc.) or found in the relevant literature (gearbox efficiency).

The moments of inertia were calculated by approximating the mass distribution for each axis and for the relevant components. The torque constants of the motors could be determined indirectly from the step response of the system and by measuring the armature current of the dc motors. The resistive torque could be determined by measuring the current needed to spin the motors without any load. Some fine tuning was also used to achieve a model response that matches the dynamics of the real robot as close as possible.

The model is validated with real data by applying the same input to both the model and the real robot. The positions and velocities are compared in the following set of plots.

*Figure 11. Comparison between the simulated robot model and the real robot.*
3.3 Sliding mode control for a WMR

3.3.1 General aspects

SMC theory [75] [76] addresses some of the most challenging problems in control. One of the most important aspects of SMC is the ability to deal with uncertain nonlinear dynamic systems. In practice, controllers are designed taking into account requirements and performance of the closed-loop system and they have to be able to meet them in all the possible circumstances which can arise during normal functioning and sometimes even in the presence of faults or disturbances. But in real applications, the engineers have to deal with uncertainties during the design phase (due to discrepancies between the model and the system) or with external disturbances and faults which
cannot be anticipated exactly. These problems have led to the development of a new branch in control theory commonly known as robust control [77]. Robust control techniques aim to design controllers that are able to deal with uncertainties and external disturbances.

Some of the most known robust control techniques are:

- **H-infinity loop shaping** [78], minimizes the sensitivity of the system over the frequency spectrum. The method is applied in the case of linear systems;
- **feedback linearisation** [79], transforms the nonlinear system into an equivalent linear one through a change of variables;
- **loop transfer recovery (LQR)**, an LQR controller is a combination of a Kalman [80] filter and a linear quadratic regulator (LQR);
- **quantitative feedback theory** [81] is a frequency domain technique that uses the Nichols chart in order to achieve a desired robust controller design.

The research field in robust control is still active and new methods are still emerging.

In the following of this chapter we will deal with one of the most known methods in robust control, sliding mode. Sliding mode deals with nonlinear systems by applying a discontinuous control law which forces the system to 'slide' along a chosen surface (i.e. a section of the system's normal behaviour).

3.3.2 Sliding mode trajectory tracking using the kinematic model of the WMR

3.3.2.1 Control law

To develop a robust controller for WMR trajectory tracking, we assume that the trajectory is previously generated by a trajectory planner. One trajectory generation method used in conjunction with a sliding mode controller is described in [82]. This method uses fifth order polynomials (quintic polynomial curves) in order to generate smooth accelerations. The sliding mode kinematic controller will use the trajectory parameters to generate the control signals.

The trajectory parameters are defined as follows:
- \( x_d, y_d, \theta_d \) - desired pose;
- \( v_d, \omega_d \) - desired linear and angular velocities;
- \( \dot{v}_d, \dot{\omega}_d \) - desired linear and angular accelerations;
The kinematic model for the non-holonomic WMR is:

\[
\begin{align*}
\dot{x}_r &= v_r \cos \theta_r \\
\dot{y}_r &= v_r \sin \theta_r \\
\dot{\theta}_r &= \omega_r
\end{align*}
\] (3.29)

where \([x_r, y_r, \theta_r]\) is the robot pose (the position of the middle of driving axis and the orientation), \(v_r\) is the forward velocity and \(\omega_r\) is the angular velocity.

The tracking error vector \([x_e, y_e, \theta_e]^T\) is defined as a transformation in robot frame:

\[
\begin{bmatrix}
x_e \\
y_e \\
\theta_e
\end{bmatrix} =
\begin{bmatrix}
\cos \theta_d & \sin \theta_d & 0 \\
-\sin \theta_d & \cos \theta_d & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_r - x_d \\
y_r - y_d \\
\theta_r - \theta_d
\end{bmatrix}
\] (3.30)

The error derivatives are:

\[
\begin{align*}
\dot{x}_e &= -v_d + v_r \cdot \cos \theta_e + y_e \omega_d \\
\dot{y}_e &= v_r \cdot \sin \theta_e - x_e \omega_d \\
\dot{\theta}_e &= \omega_r - \omega_d
\end{align*}
\] (3.31)

The following surfaces are proposed:

\[
\begin{align*}
s_1 &= \dot{x}_e + k_1 x_e \\
s_2 &= \dot{\theta}_e + k_2 \theta_e + k_0 y_e
\end{align*}
\] (3.32)

where \(k_0, k_1, k_2\) are positive defined parameters of the sliding surfaces.
According to Gao and Hung [83], the control law can be derived from:

$$\dot{s} = -Q \cdot s - P \cdot sgn(s)$$  \hspace{1cm} (3.33)

where

$$s = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}.$$

Surfaces and control law stability can be proven using Lyapunov stability.

A Lyapunov function candidate is chosen as:

$$V = \frac{1}{2} s^T s$$

$$\dot{V} = s_1 \dot{s}_1 + s_2 \dot{s}_2$$

$$\dot{V} = s_1 (-Q_1 s_1 - P_1 sgn(s_1)) + s_2 (-Q_2 s_2 - P_2 sgn(s_2))$$

$$\dot{V} = -Q_1 s_1^2 - Q_2 s_2^2 - P_1 |s_1| - P_2 |s_2|$$

In order for $\dot{V}$ to be negative semi-definite we have to choose:

$$Q_1 \geq 0, Q_2 \geq 0, P_1 \geq 0, P_2 \geq 0$$  \hspace{1cm} (3.34)

Control surfaces stability is proven under the assumption that $s_1 = 0$ and $s_2 = 0$.

If $s_1 = 0 \Rightarrow \dot{x}_e = -k_1 x_e$ implies $x_e \to 0$.

If $s_2 = 0 \Rightarrow \dot{\theta}_e = -k_2 \theta_e - k_0 \gamma_e$

We assume that $x_e$ goes to zero first which leads to $\dot{\gamma}_e = \nu_r \cdot \sin \theta_e$ and we have the following cases:

1. if $\gamma_e > 0$ and $\theta_e > 0 \Rightarrow \dot{\theta}_e < 0$ which means that $\theta_e$ is decreasing until $\theta_e = -\gamma_e$. For $\theta_e < 0 \Rightarrow \gamma_e < 0$ (assuming $\nu_r$ is positive in steady state) and both $\gamma_e$ and $\theta_e$ will converge to zero.

2. $\gamma_e < 0$ and $\theta_e < 0$, $\theta_e$ is increasing until $\theta_e = -\gamma_e$. For $\theta_e < 0 \Rightarrow \dot{\gamma}_e < 0$ and $\gamma_e$ and $\theta_e$ will converge together to zero.

Next we can derive the control law.

$$\dot{s}_1 = \ddot{x}_e + k_1 \dot{x}_e = \nu_r \cdot \cos \theta_e + v_r \dot{\theta}_e \sin \theta_e + \gamma_e \omega_d + \gamma_e \dot{\omega}_d - \dot{\nu}_d + k_1 \dot{x}_e$$  \hspace{1cm} (3.35)

$$\dot{s}_2 = \dot{\theta}_e + k_2 \dot{\theta}_e + k_0 \dot{\gamma}_e = \dot{\omega}_r - \dot{\omega}_d + k_2 \dot{\theta}_e + k_0 \dot{\gamma}_e$$

The control signals for the non-holonomic robot are $\nu_r$ and $\omega_r$:

$$\dot{\nu}_c = -Q_1 s_1 - P_1 sgn(s_1) - v_r \dot{\theta}_e \sin \theta_e - \gamma_e \omega_d - \gamma_e \dot{\omega}_d - \dot{\nu}_d - k_1 \dot{x}_e$$  \hspace{1cm} (3.36)

$$\dot{\omega}_c = -Q_2 s_2 - P_2 sgn(s_2) + \dot{\omega}_d - k_2 \dot{\theta}_e - k_0 \dot{\gamma}_e \dot{s}_2 = \ddot{\theta}_e + k_2 \dot{\theta}_e + k_0 \dot{\gamma}_e$$

$$= \dot{\omega}_r - \dot{\omega}_d + k_2 \dot{\theta}_e + k_0 \dot{\gamma}_e$$
After integration, we get the control signals for the non-holonomic robot, i.e., linear and angular velocities of the robot. These can be transformed in control angular velocities for left and right wheel respectively.

\[
\omega_{cR} = \frac{v_c + L\omega_c}{R_R} \\
\omega_{cL} = \frac{v_c - L\omega_c}{R_L}
\]  

(3.37)

The following bloc diagram illustrates the cascade control for the non-holonomic robot.

![Block diagram of the control structure with the kinematic SMC and PI controllers for the wheels](image)

**Figure 14. Block diagram of the control structure with the kinematic SMC and PI controllers for the wheels**

The kinematic SMC is used for trajectory tracking and it generates the control velocities for the wheels. The DC motors for each wheel (the actuators) are then controlled using two PI controllers. In this case, the dynamics of the robot are managed by the PI controllers.

The cascade control offers some advantages when compared to a single controller approach. For instance, in this case, the inner loop (PI controller) has a faster response and can compensate quicker for disturbances. The outer loop, sliding mode controller, can run at a slower rate which is influenced by various factors (trajectory planner, processing power). Hence, decoupling the controllers creates a more stable and robust control strategy. For this particular case the sampling time for the inner control loop was 0.02s and for the outer loop was 0.15s.

### 3.3.2.2 Test results

The controller was first tested and tuned in simulation using the model developed in Section 3.2 (see Figure 15). The parameters validated in simulation were then applied to the real robot and the behaviour of the controller is shown Figure 16. The robot follows the desired trajectory with minimal errors and chattering.
Figure 15. Trajectory tracking results for the kinematic SMC tested in simulation
3.3.3 Sliding mode trajectory tracking using the dynamic model

3.3.3.1 Control law

Although the most common model used for developing the SMC for a WMR is the kinematic model in some cases the dynamic model can be useful because it offers improved control of the dynamics of the close loop system. The dynamic model is needed especially when dealing with uncertainties or faults in the parameters of the system. In our case, we aim to develop a fault tolerant SMC which is robust to variations in the wheel radius. Because the variations can occur in each wheel the dynamic model will have to deal with them separately.

In the following, the SMC will be derived starting from a simplified version of the Lagrange dynamic model.

The general form of the Lagrange dynamic model for a WMR:

\[
M(q) \ddot{q} + V(q, \dot{q}) \dot{q} + \tau_d = B(q) \tau - F_R(\dot{q}) - \Lambda^T(q) \tag{3.38}
\]

The simplified version we will use for deriving the control law:

\[
\begin{bmatrix}
m & 0 & 0 \\
0 & m & 0 \\
0 & 0 & J
\end{bmatrix}
\begin{bmatrix}
\ddot{x}_r \\
\ddot{y}_r \\
\ddot{\theta}_r
\end{bmatrix}
= \begin{bmatrix}
\cos \theta_r & \cos \theta_r \\
\frac{R_R}{\sin \theta_r} & \frac{R_L}{\sin \theta_r} \\
\frac{R_R}{L} & \frac{R_L}{-L}
\end{bmatrix}
\begin{bmatrix}
\tau_R \\
\tau_L \\
\tau_r
\end{bmatrix}
+ \begin{bmatrix}
\sin \theta_r \\
-\cos \theta_r \\
0
\end{bmatrix}
\times \lambda \tag{3.39}
\]

Next, we can use:

\[
\lambda = -m \cdot (\dot{x}_r \cdot \cos \theta_r + \dot{y}_r \cdot \sin \theta_r) \cdot \dot{\theta}_r \tag{3.40}
\]
And the non-holonomic constraint:

\[ \dot{x}_r \cdot \sin \theta_r - \dot{y}_r \cdot \cos \theta_r = 0 \]  
(3.41)

The dynamic model can be written explicitly:

\[
\begin{align*}
\ddot{x}_r &= \left( \frac{\tau_R}{R_R} + \frac{\tau_L}{R_L} \right) \cdot \frac{\cos \theta_r}{m} - \dot{y}_r \dot{\theta}_r \\
\ddot{y}_r &= \left( \frac{\tau_R}{R_R} + \frac{\tau_L}{R_L} \right) \cdot \frac{\sin \theta_r}{m} + \dot{x}_r \dot{\theta}_r \\
\ddot{\theta}_r &= \left( \frac{\tau_R}{R_R} - \frac{\tau_L}{R_L} \right) \cdot \frac{L}{J}
\end{align*}
\]  
(3.42)

Before we can derive the sliding mode control law the wheel radius uncertainty is written as:

\[
\begin{align*}
R_{Rreal} &= R_{Rnom} + \Delta R_R \\
R_{Lreal} &= R_{Lnom} + \Delta R_L
\end{align*}
\]  
(3.43)

The following surfaces will be used:

\[
\begin{align*}
s_1 &= \dot{x}_e + k_1 x_e \\
s_2 &= \dot{\theta}_e + k_2 \theta_e + k_0 y_e
\end{align*}
\]  
(4.44)

The control law can be derived from:

\[
\begin{align*}
\dot{s}_1 &= \dot{x}_e + k_1 \dot{x}_e = -Q_1 \cdot s_1 - P_1 \cdot sgn(s_1) \\
\dot{s}_2 &= \dot{\theta}_e + k_2 \dot{\theta}_e + k_0 \dot{y}_e = -Q_2 \cdot s_2 - P_2 \cdot sgn(s_2)
\end{align*}
\]  
(3.45)

where:

\[
\begin{align*}
\dot{x}_e &= \dot{x}_r \cos \theta_d - \dot{x}_r \dot{\theta}_d \sin \theta_d + \dot{y}_r \sin \theta_d + \dot{y}_r \dot{\theta}_d \cos \theta_d + \dot{y}_e \omega_d + y_e \dot{\omega}_d - \dot{v}_d \\
\dot{y}_e &= -\dot{x}_r \sin \theta_d - \dot{x}_r \dot{\theta}_d \cos \theta_d + \dot{y}_r \cos \theta_d - \dot{y}_r \dot{\theta}_d \sin \theta_d - \dot{x}_e \omega_d - x_e \dot{\omega}_d \\
\dot{\theta}_e &= \dot{\omega}_r - \dot{\omega}_d
\end{align*}
\]  
(3.46)

After some manipulation we get the control torques for the right and left wheels:

\[
\begin{align*}
\tau_{cR} &= \frac{R_{Rnom} + \Delta R_R}{2} \cdot \left[ \frac{m}{\cos \theta_e} \cdot (-Q_1 \cdot s_1 - P_1 \cdot sgn(s_1) + T_1) + \frac{J}{L} \cdot (-Q_2 \cdot s_2 - P_2 \cdot sgn(s_2) + T_2) \right] \\
\tau_{cL} &= \frac{R_{Lnom} + \Delta R_L}{2} \cdot \left[ \frac{m}{\cos \theta_e} \cdot (-Q_1 \cdot s_1 - P_1 \cdot sgn(s_1) + T_1) - \frac{J}{L} \cdot (-Q_2 \cdot s_2 - P_2 \cdot sgn(s_2) + T_2) \right]
\end{align*}
\]
where:
\[ T_1 = \dot{x}_e \dot{\theta}_e \sin \theta_d + \dot{y}_e \dot{\theta}_e \cos \theta_d - \dot{y}_e \omega_d - y_e \dot{\omega}_d + \ddot{v}_d - k_1 \ddot{x}_e \]
\[ T_2 = \dot{\omega}_d - k_2 \dot{\theta}_e - k_3 \dot{y}_e \]

In practice, it is not always possible to send the control torque directly to the actuators of the WMR. The actuators are usually DC motors which are not so easy to control in torque. The control torques can be transformed in linear and angular accelerations for the robot:

\[
\dot{v}_c = \frac{1}{m} \left( \frac{\tau_R}{R_R} + \frac{\tau_L}{R_L} \right)
\]
\[
\dot{\omega}_c = \frac{L}{J} \left( \frac{\tau_R}{R_R} - \frac{\tau_L}{R_L} \right)
\]

In this case, the controls become very similar to the controls of the kinematic SMC (they are not identical because of the terms \( \Delta R_R \) and \( \Delta R_L \)).

We assume that \( R_{rnom} = R_{lnom} \) and \( \Delta R_R = \Delta R_L \), then the accelerations of the mobile robot become:

\[
\dot{v}_{cd} = \frac{R_{rnom} + \Delta R}{R_{rnom}} \dot{v}_{ck}
\]
\[
\dot{\omega}_{cd} = \frac{R_{rnom} + \Delta R}{R_{rnom}} \dot{\omega}_{ck}
\]

If \( \Delta R = 0 \) the dynamical model becomes identical with the kinematic one. The dynamic SMC is redundant in this situation and the kinematic SMC should be used instead. The dynamics of the robot would be handled by the PI controller of the dc motor for each wheel.

An option to avoid using the PI controller is to add the DC motor model to the SMC and convert the control torques to control voltages for the right and left wheels. Although this approach works in simulations with models without friction, resistive torque or Coriolis forces, it fails to converge to zero when the control is applied to a real robot.

To allow the dynamic SMC to handle the dynamics of the WMR another controller needs to be developed for controlling the torque of the dc motors. The output torque produced by a dc motor is proportional with its armature current. This means that a torque controller can be implemented by adding current sensors on the dc motor for each wheel. However, this approach has the same drawbacks previously described for the first approach which means a different type of SMC needs to be developed for handling system dynamics.
3.3.3.2 Test results

The tests on the robot were performed using the dynamic SMC coupled with the PI controllers for each wheel. Figure 18 illustrates the behaviour of the system when $\Delta R = 0.04$ and there is no fault present.
Figure 18. Test results when the robot has no fault

Figure 19 shows the behaviour of the system when the left wheel radius changes from $R_L = 0.1145$ to $R_L = 0.08$. There is a slight increase in the tracking errors when the fault occurs, but the controller manages to compensate and maintain the desired trajectory.
Figure 19. Test results when the robot has a fault (flat tyre) and $\Delta R = 0.04m$

Next, the controller was tested with $\Delta R = 0$ (Figure 20) which means that the performance should be the same as in the case of kinematic SMC because the dynamics are completely cancelled.
in this case. The controller is still able to follow the trajectory, but tracking performance is poor. Big errors can be observed, especially in the tracking angle.

Figure 20. Test results when the robot has a fault and $\Delta R = 0$
Figure 21. Comparison between the angular velocities of the wheels:

a) Without fault; b) With fault in the left wheel (flat tyre).

The following set of plots shows the behaviour of the robot when torque control is used instead of velocity control for the dc motors. The controller does not follow the trajectory properly because of uncertainties in the model (friction, resistive torque, etc.). The first set of plots (Figure 22) presents the results obtained with the robot model in simulation, while the second set of plots (Figure 23) shows the behaviour of the real robot when the same parameters were used.
Figure 22. Test results when using torque control for the dynamic SMC in simulation

Figure 23. Test results when using torque control for the dynamic SMC with the real robot
3.3.4 Sliding mode trajectory tracking using kinematic and dynamic controllers

3.3.4.1 Control law

As it was shown in Section 3.3.3, it is not possible to use the dynamic SMC and send a control torque to the wheels. Using the dynamic SMC in combination with a PI controller is also not an option because the dynamics of the SMC model are cancelled (they are handled by the PI controller).

An alternative way to tackle this problem is to use a combination of two SMC controllers, one for the trajectory tracking (kinematic controller) and another SMC which incorporates the dynamic model and controls the DC motors of the wheels. In this approach, the dynamic SMC replaces the PI controller in the inner loop.

The dynamic model can be written with acceleration as outputs and torques as inputs:

\[
\begin{align*}
\dot{v}_c &= \frac{1}{m} \cdot \left( \frac{\tau_R}{R_R} + \frac{\tau_L}{R_L} \right) \\
\dot{\omega}_c &= \frac{L}{J} \cdot \left( \frac{\tau_R}{R_R} - \frac{\tau_L}{R_L} \right)
\end{align*}
\] (3.49)

The velocity errors are defined as:

\[
\begin{align*}
v_e &= v_r - v_d \\
\omega_e &= \omega_r - \omega_d
\end{align*}
\] (3.50)

We chose the following surfaces for the SMC as proposed in [26]:

\[
\begin{align*}
s_1 &= v_e + k_1 \int_0^t v_e(\tau) d\tau \\
s_2 &= \omega_e + k_2 \int_0^t \omega_e(\tau) d\tau
\end{align*}
\] (3.51)

\( s_1 = 0 \) gives \( v_e = -k_1 \int_0^t v_e(\tau) d\tau \) which leads to \( v_e \to 0 \) because \( k_1 > 0 \). The same applies for \( s_2 = 0 \) which leads to \( \omega_e \to 0 \).

Next, the control law can be derived using: \( \dot{s} = -Q \cdot s - P \cdot sgn(s) \).

\[
\begin{align*}
\dot{s}_1 &= \dot{v}_e + k_1 v_e \\
\dot{s}_2 &= \dot{\omega}_e + k_2 \omega_e
\end{align*}
\] (3.52)
After some manipulations we get:

\[
\tau_{cR} = \frac{R_{Rnom} + \Delta R_R}{2} \left( m \cdot (\dot{v}_d - k_1 v_e - Q_1 \cdot s_1 - P_1 \cdot sgn(s_1)) + \frac{J}{L} \right) \cdot \left( \dot{\omega}_d - k_2 \omega_e - Q_2 \cdot s_2 - P_2 \cdot sgn(s_2) \right)
\]

\[
\tau_{cL} = \frac{R_{Lnom} + \Delta R_L}{2} \left( m \cdot (\dot{v}_d - k_1 v_e - Q_1 \cdot s_1 - P_1 \cdot sgn(s_1)) - \frac{J}{L} \right) \cdot \left( \dot{\omega}_d - k_2 \omega_e - Q_2 \cdot s_2 - P_2 \cdot sgn(s_2) \right)
\]

The control torques can be converted into control voltages for the DC motor by using the DC motor model:

\[
J \ddot{\theta} + b \omega = K_te I
\]

\[
L_e \frac{dI}{dt} + R_e I = V - K_te \omega
\]

In steady state, we neglect the influence of the inductance \( L_e \), hence we get:

\[
V_{cR} = \frac{\tau_{cR} R_e + K_te \omega_R}{K_te}
\]

\[
V_{cL} = \frac{\tau_{cL} R_e + K_te \omega_L}{K_te}
\]

Using this controller, we can deal with uncertainty or faults not only in the robot body but also in the actuators (right and left DC motors).
3.3.4.2 Simulation data

For this controller, tests were conducted using the simulated robot model because this allows observing the controller response when a sudden fault appears, which will be impossible to test on the real system. As it can be seen from the plots, the overall performance of the two SMC structure is very good. Dynamic performance is good, tracking errors are small and the controller quickly compensates for the presence of a fault.

Figure 25. Simulation results for the kinematic SMC with separate dynamic SMC

The control signals are shown in Figure 26. The fault in the radius of the right wheel is compensated with the control torque by the dynamic SMC. The effect of the fault can also be seen in the control signals of the kinematic controller, but they return to the desired values after the dynamic SMC manages to overcome the fault.
Although the controller shows robustness to plant uncertainty and faults the control surfaces do not go to zero as they should. In [84] an adaptive control law was proposed to solve this problem. In the following section, an approach which uses different surfaces will be proposed in order to achieve the sliding mode without an adaptive control law.
3.3.5 Sliding mode trajectory tracking using kinematic and dynamic controllers with torque derivative control

3.3.5.1 Control law

The model and velocity errors are defined as:

\[ \dot{v}_c = \frac{1}{m} \left( \frac{\tau_R}{R_R} + \frac{\tau_L}{R_L} \right) \] (3.55)

\[ \dot{\omega}_c = \frac{L}{J} \left( \frac{\tau_R}{R_R} - \frac{\tau_L}{R_L} \right) \]

\[ v_e = v_r - v_d \] (3.56)

\[ \omega_e = \omega_r - \omega_d \]

We chose the following surfaces:

\[ s_1 = \dot{v}_e + k_1 v_e \] (3.57)

\[ s_2 = \dot{\omega}_e + k_2 \omega_e \]

\( s_1 = 0 \) gives \( \dot{v}_e = -k_1 v_e \) which leads to \( v_e \to 0 \) because \( k_1 > 0 \). The same applies for \( s_2 = 0 \) which leads to \( \omega_e \to 0 \).

The control is derived using: \( \dot{s} = -Q \cdot s - P \cdot \text{sgn}(s) \):

\[ \dot{s}_1 = \ddot{v}_e + k_1 \dot{v}_e \] (3.58)

\[ \dot{s}_2 = \ddot{\omega}_e + k_2 \dot{\omega}_e \]

After some manipulations we get:

\[ \tau_{cr} = \frac{R_{R\text{nom}} + \Delta R_R}{2} \left( m \cdot (\ddot{v}_d - k_1 \dot{v}_e - Q_1 \cdot s_1 - P_1 \cdot \text{sgn}(s_1)) + \frac{J}{L} \cdot (\ddot{\omega}_d - k_2 \dot{\omega}_e - Q_2 \cdot s_2 - P_2 \cdot \text{sgn}(s_2)) \right) \] (3.59)

\[ \tau_{cl} = \frac{R_{L\text{nom}} + \Delta R_L}{2} \left( m \cdot (\ddot{v}_d - k_1 \dot{v}_e - Q_1 \cdot s_1 - P_1 \cdot \text{sgn}(s_1)) - \frac{J}{L} \cdot (\ddot{\omega}_d - k_2 \dot{\omega}_e - Q_2 \cdot s_2 - P_2 \cdot \text{sgn}(s_2)) \right) \]

In this approach, the control torque derivatives are given as control signals by the SMC. This is desirable because the integration of the control law will lead to chattering attenuation, but the most
important advantage is that the control surfaces will converge to zero (which was a problem for the control strategy described in Section 3.3.4).

### 3.3.5.2 DC motor torque control using sliding mode

The dynamic SMC derived previously has a torque control signal as an output. Usually, DC motor control is made using angular velocity as output, but in the case of the dynamic SMC, this is not an option because it cancels the dynamics of the controller. To overcome this and to study the effects of faults in the actuators (i.e. the DC motors) an SMC for the DC motor armature current will be derived.

The DC motor model:

\[ J \ddot{\theta} + b \omega = K_t e I \]  
\[ L_e \frac{dI}{dt} + R_e I = V - K_t e \omega \]  

Output torque is proportional with the armature current, thus the error will be defined as:

\[ I_e = I_m - I_d \]  

The surface:

\[ J_s \dot{s} = I_e + k I_e \]  

The control can be derived using: \( \dot{s} = -Q \cdot s - P \cdot sgn(s) \) and the DC motor model:

\[ V_c = \dot{I}_d + (R_e + \Delta R_e) I_m + K_t e \dot{\omega} - L_e (Q \cdot s + P \cdot sgn(s) + k I_e) \]  

*Figure 28. Block diagram of the kinematic SMC with dynamic SMC and current SMC for the DC motors*
3.3.5.3 Test results

The first set of plots shows the behaviour of the robot when the left wheel radius changes from $R_L = 0.1145$ to $R_L = 0.08$.

Figure 29. Test results when the robot has a fault in the left wheel (flat tyre)
Compared with the controller presented in Section 3.3.4, in this case, the control surfaces converge to zero even in the presence of a fault.

Figure 30. Control torques generated by the dynamic SMC and armature current for each DC motor

3.4 A fault hiding approach for the sliding mode fault-tolerant control of a non-holonomic mobile robot

Several types of sliding mode controllers were studied in the previous section. A dynamical SMC was derived with the intention of dealing with faults in the robot’s wheels (flat tyre). Because this type of fault leads to a variation of the wheel radius, the dynamical model used for the derivation of the SMC had to take this variation into account. Furthermore, it was shown that for most robots, when using an angular velocity controller for the wheels, the dynamics of the robot are not handled by the SMC. Taking this into consideration, a new active Fault Tolerant Control (FTC) technique was developed instead of the passive one used for designing the dynamic SMC.

By using fault detection, the dynamical SMC presented in Section 3.3 is replaced with the simpler kinematic SMC in conjunction with a fault hiding block. This fault hiding technique is very appealing and has been investigated by many researchers in the last years. In this approach, because the fault is hidden from the controller, retuning is not needed which leads to time and cost savings.

A paper with the same title as the current section was written and submitted for publication. The most relevant aspects presented in the paper are summarised in the following sections.
3.4.1 Including a fault hiding mechanism

The previous SMC control scheme presented in Section 3.3.2 can be adapted to handle faults. Here, for illustrative purposes and without reducing the generality of the proposed fault-tolerant control method, the case of a fault affecting the wheel radius (flat tyre) is considered. There are two ways of including the faults in the SMC control scheme: the SMC could be readjusted on-line, or alternatively, a fault hiding approach can be used without the need of retuning the controller. Here, the second option is the one selected. In Figure 31, the SMC scheme proposed in Figure 14 is modified by including a block which contains the fault hiding mechanism for each of the active wheels of the mobile robot. The role of this block is to hide the effect of fault to the SMC avoiding to retune it.

![Figure 31. Fault hiding for the sliding mode fault-tolerant control of a nonholonomic mobile robot for fault in the actuators (flat tyre).](image)

In the following proposition, we will show that the fault hiding mechanism is of the form \( \frac{\Delta R}{R_{\text{nom}}} \hat{v}_c \) and \( \frac{\Delta R}{R_{\text{nom}}} \hat{\omega}_c \), where \( R_{\text{nom}} \) is the nominal radius of the wheel and \( \Delta R \) is the maximum range of radius variation. Moreover, it will be shown that the sliding mode controller based on the kinematic model together with the fault hiding block is stable.

**Proposition 3.1:** Given a robot with two active wheels, the following fault accommodation mechanism

\[
\begin{align*}
\hat{v}_{\text{fault}} &= \frac{R_{\text{nom}} + \Delta R}{R_{\text{nom}}} \hat{v}_c \\
\hat{\omega}_{\text{fault}} &= \frac{R_{\text{nom}} + \Delta R}{R_{\text{nom}}} \hat{\omega}_c
\end{align*}
\]  

(3.64)

is stable and robust against faults located in the mobile robots wheels.
Proof:

**Remark 3.1:** If $\Delta R = 0$, the derivatives of linear and angular velocities of the robot obtained using SMC based on the simplified dynamic model becomes identical with the ones obtained using SMC based on the kinematic model, i.e., $\dot{v}_{\text{fault}} = \dot{v}_c$ and $\dot{\omega}_{\text{fault}} = \dot{\omega}_c$.

Considering the faults which can occur in the wheels of the mobile robot (flat tyre) we can distinguish the following three cases.

**Case 1.** Both wheels are faulty:

\[
R_{Rf} < R_{nom} \text{ and } (R_{nom} - R_{Rf}) < \Delta R \\
R_{Lf} < R_{nom} \text{ and } (R_{nom} - R_{Lf}) < \Delta R
\]  

(3.65)

The angular velocities for each wheel will be:

\[
\omega_{CRf} = \frac{v_{faul} + L\omega_{faul}}{R_{nom}}
\]  

(3.66)

\[
\omega_{CLf} = \frac{v_{faul} - L\omega_{faul}}{R_{nom}}
\]

where $R_{Rf}$ and $R_{Lf}$ are the radius of the right and left faulty wheels, respectively. $\omega_{CRf}$ and $\omega_{CLf}$ are the angular velocities of the right and left faulty wheels, respectively.

**Case 2.** Left wheel is faulty, right wheel is non-faulty:

\[
R_{Rf} = R_{nom} \\
R_{Lf} < R_{nom} \text{ and } (R_{nom} - R_{Lf}) < \Delta R
\]  

(3.67)

The angular velocities for each wheel will be:

\[
\omega_{CRf} = \frac{v_c + L\omega_c}{R_R}
\]  

(3.68)

\[
\omega_{CLf} = \frac{v_{faul} - L\omega_{faul}}{R_{nom}}
\]

**Case 3.** Right wheel is faulty, left wheel is non-faulty:

\[
R_{Rf} < R_{nom} \text{ and } (R_{nom} - R_{Rf}) < \Delta R \\
R_{Lf} = R_{nom}
\]  

(3.69)

The angular velocities for each wheel will be:

\[
\omega_{CRf} = \frac{v_{faul} + L\omega_{faul}}{R_{nom}}
\]  

(3.70)

\[
\omega_{CLf} = \frac{v_c - L\omega_c}{R_L}
\]
3.4.2 Results

In this section, after the validation of the mobile robot model (presented in Section 3.2), simulated results of the proposed fault-tolerant control method are presented in fault-free scenarios and faulty scenarios. The simulation is used in order to be able to “inject” faults in the wheel at any time instant. Using a realistic model, it is possible to analyse the proposed fault-tolerant control strategy for faults which appear at any time instant. This is not possible for the real robot in which case the fault is considered to be present before we start the experiment.

3.4.2.1 Simulation results of SMC based on the kinematic model in non-faulty case

Design parameters of the chosen surfaces and control law were chosen in accordance with (3.34) in order to guarantee the convergence of the error by keeping the surfaces \( s_1 \) and \( s_2 \) to zero (sliding phase) and to guarantee the convergence of the control law (reaching phase). Moreover, in order to reduce the chattering phenomenon, the signum functions in the control law (3.33) were replaced by saturation functions with the threshold \( \pm 0.5 \). Figure 32a–g shows the results for trajectory-tracking using SMC based on the kinematic model in fault-free scenario. It can be seen that linear and angular velocities of the mobile robot (Figure 32c and Figure 32d) follow the references generated a priori by the Trajectory Planner. The simulated mobile robot is able to track with small errors the virtual robot (see Figure 32a). It is also shown that the sliding surfaces \( s_1 \) and \( s_2 \) are kept to zero (Figure 32e and Figure 32f). Consequently (see Proposition 3.1), the tracking errors converge to zero Figure 32b, as well. In Figure 32h the angular velocities for both wheels right and left are shown. These angular velocities are maintained by means of a simple PI controller as it was shown in Figure 31.

![Image of results graphs](image-url)
Figure 32. Simulation results of trajectory tracking using an SMC based on a kinematic model in a non-faulty situation
3.4.2.2 Simulation results of SMC based on the kinematic model in faulty case

The fault is introduced in the left wheel (flat tire of the left wheel). The radius of the left wheel is changed from \( R_L = R_{\text{nom}} = 0.1145 \) meters to \( R_{Lf} = 0.0745 \) meters and it is “injected” to the robot model at \( t = 16 \) seconds and will be faulty for the rest of the simulation. In this case, we do not consider the fault hiding block and the trajectory-tracking is based on the control strategy proposed in Section 3.3.2 (see Figure 29), i.e., SMC based on the kinematic model of the mobile robot.

In Figure 33a one can observe the tracking errors at the moment when the fault occurs (\( t = 16 \) seconds). After a finite time, the control law manages to make both sliding surfaces zero (Figure 33d and Figure 33e) and consequently, the tracking errors converge to zero (Figure 33a). In Figure 33g it can be seen that the angular velocity of the faulty wheel (dotted red line) increases as a result of a big control effort needed to overcome the fault effect (the control velocities observed in Figure 33f). In Figure 33b and Figure 33c, it can be seen the real linear and angular velocities of the robot and their variations when the fault occurs. The SMC based on the kinematic model is robust even in the presence of the fault in the left wheel thanks to the robustness properties of the SMC strategy. Since the fault can be of any magnitude, we cannot guarantee the robustness against any possible faults. In order to deal with this situation we have, somehow, to take into account during the controller design phase, the fault magnitude and to prove the robustness of the fault tolerant controller against a maximum possible fault magnitude (\( \Delta R \)). In this way, we can guarantee the robustness of the fault-tolerant controller (SMC plus fault hiding block) for all possible faults with magnitudes less or equal than the maximum one considered.
Figure 33. Simulation results of trajectory tracking using an SMC based on a kinematic model in a faulty situation without fault hiding block

Now, we will apply the fault hiding strategy proposed in Section 3.4.1, see Figure 31. Here, we consider the maximum range of radius variation to be $\Delta R = 0.06$ meters. From Figure 34a–f, it can be seen that because of the fault hiding approach is considered, the SMC based on the kinematic
model is almost insensitive to the fault in the left wheel, occurred at $t = 16$ seconds, comparing with the sensitivity in the case of SMC without the fault hiding block as it was shown in Figure 34a–f. In Figure 34f, it can be seen that after just a few iterations (the time need for the fault hiding block to become active), the control signals (linear and angular velocities of the mobile robot) have the same values as in the case of fault free situation (Section 3.4.2.1), Figure 32 g). The item responsible of the change in the angular velocity for the left wheel (dotted red in Figure 34g) after the fault occurrence is the fault hiding block and not the SMC which remains insensitive for the rest of the simulation (see Figure 34f).
3.5 Sliding mode control of a WMR – interval analysis approach

In the previous sections, various types of SMC with different degrees of complexity were designed and tested. Although the sliding mode control strategy is robust against model uncertainties and disturbances, there is no analytical method for tuning the parameters or to analyse the degree of robustness. Some recommendations for experimental methods of tuning are presented in [3]. Furthermore, most of the SMC related publications mentions that SMC can be robust for small uncertainties in the model or small disturbances. The degree of robustness is not quantifiable and the effect of cumulative uncertainties or disturbances is not studied. To overcome these drawbacks a new approach based on interval analysis is proposed.
To study the stability of the SMC we can distinguish several cases based on the place where the uncertainty or disturbance can appear in the control loop:

- uncertainty in the measurements – in most cases, measurements uncertainty is small, but if the SMC has a complex control law the cumulative effect of these uncertainties can decrease the robustness of the controller;
- uncertainty in the mathematical model of the system – a mathematical model of the control system is needed for developing the SMC control law. The complexity and accuracy of the model has a direct impact on the robustness of the control loop;
- uncertainty in the control output – the control output computed by the SMC can be applied directly to actuators or as a set-point to an inner control loop when the control system uses a cascade control strategy. When cascade control is used, the inner loop has its own dynamics, which means that the control action demanded by the SMC from the outer loop will always have some uncertainty that affects the robustness of the control loop.

The influence of uncertainties on the stability of SMC is going to be studied for a WMR in the current section. Uncertainties in the measurements and uncertainties in the control output are studied using the SMC with the kinematic model described in Section 3.3.2, while uncertainties in the mathematical model of the system are going to be studied on the SMC with the dynamic model described in Section 3.3.3.

3.5.1 Stability analysis of an SMC with uncertainty in the velocity measurements

With reference to Section 3.3.2 Equation (3.29) the kinematic model with uncertainty in the velocities can be written as follows:

\[
\begin{align*}
\dot{x}_r &= (v_r + p_1)\cos \theta_r \\
\dot{y}_r &= (v_r + p_1)\sin \theta_r, \\
\dot{\theta}_r &= \omega_r + p_2
\end{align*}
\]  

(3.71)

where \([x_r, y_r, \theta_r]\) is the robot pose, \([v_r, \omega_r]\) are the linear and angular velocities of the robot and \(p_1 \in [p_1, \bar{p}_1], p_2 \in [p_2, \bar{p}_2]\) are their corresponding measurement uncertainties.

The tracking errors derivatives are:

\[
\begin{align*}
\dot{x}_e &= -v_d + (v_r + p_1) \cdot \cos \theta_e + y_e \omega_d \\
\dot{y}_e &= (v_r + p_1) \cdot \sin \theta_e - x_e \omega_d \\
\dot{\theta}_e &= \omega_r - \omega_d
\end{align*}
\]  

(3.72)
We chose the following sliding surfaces (recall Section 3.3.2 Equation (3.32)):

\[ s_1 = \dot{x}_e + k_1 x_e \]  
\[ s_2 = \dot{\theta}_e + k_2 \theta_e + k_0 y_e \]  

(3.73)

And the control law:

\[ \dot{s} = -Q \cdot s - P \cdot sgn(s), \quad \text{where} \quad s = [s_1 s_2]^T \]

For the model without uncertainty \( \dot{s} \) has the following form:

\[ \dot{s} = \begin{pmatrix} \dot{s}_1 \\ \dot{s}_2 \end{pmatrix} = \begin{pmatrix} v_r \cdot \cos \theta_e - v_r \dot{\theta}_e \sin \theta_e + \dot{y}_e \omega_d + y_e \omega_d - \dot{v}_d + k_1 x_e \\ \dot{\omega}_r - \dot{\omega}_d + k_2 \dot{\theta}_e + k_0 y_e \end{pmatrix} \]  

(3.74)

After some derivation, we obtain \( \dot{s} \) with the velocity uncertainty included:

\[ \dot{s}_1 = \dot{v}_r \cdot \cos \theta_e - (v_r + p_1)(\omega_r + p_2 - \omega_d)\sin \theta_e + (v_r + p_1)\omega_d \sin \theta_e - x_e \omega_d^2 + y_e \omega_d - \dot{v}_d \\ + k_1(-v_d + (v_r + p_1) \cdot \cos \theta_e + y_e \omega_d) \\ \dot{s}_2 = \dot{\omega}_r - \dot{\omega}_d + k_2(\omega_r + p_2 - \omega_d) + k_0(v_r + p_1) \cdot \sin \theta_e - k_0 x_e \omega_d \]

The control law can now be written with the uncertainty included:

\[ \dot{s} = -Q \cdot s - P \cdot sgn(s) + \psi(x_e, y_e, \theta_e, p) \]  

where \( \psi(x_e, y_e, \theta_e, p) \) is the uncertainty function of the control law which contains all the uncertain terms:

\[ \psi = \begin{pmatrix} \psi_1 \\ \psi_2 \end{pmatrix} = \begin{pmatrix} k_1 p_1 \cos \theta_e + (2 p_1 \omega_d - p_1 \omega_r - v_r p_2 - p_1 p_2) \sin \theta_e \\ k_0 p_1 \sin \theta_e + k_2 p_2 \end{pmatrix} \]  

(3.76)

It is obvious that \( \psi = 0 \) when \( p = 0 \). We have to bound \( \psi \) for all feasible \( x_e, y_e, \theta_e \) which belong to some intervals \([x_e], [y_e], [\theta_e] \). An easy way to bound the uncertainty function is to use forward evaluation with intervals. This approach can give an overestimated result if the function has multiple occurrences of its variables in the expression. In this case \([x_e], [y_e], [\theta_e] \) have to be bisected to decrease the overestimation.

After \( \psi \) is bounded \( \dot{s} \) becomes a differential inclusion:

\[ \dot{s} \in -Q \cdot s - P \cdot sgn(s) + [\psi](x_e, y_e, \theta_e, [p]) \]  

(3.77)
To study the stability of this differential inclusion we have to use V-stability, see Definition 2.2 in Chapter 2 and [20] for reference, where the following Lyapunov candidate was proposed:

\[ V(s) = s^T s - r^2 \]  

(3.78)

We have to find \( r \) such that 3.78 becomes a Lyapunov function, i.e. \( \dot{V}(s) < 0 \). In this case, the system does not converge to 0, but instead, it converges to a region of stability defined by a circle with radius \( r \). This stability concept is weaker than the Lyapunov stability, but it is necessary for systems with uncertainties. The radius \( r \) needs to be as small as possible for a greater stability margin of the sliding mode controller.

After derivation with respect to \( s \) we get:

\[ \dot{V}(s) = 2(s_1 \dot{s}_1 + s_2 \dot{s}_2) \]  

(3.79)

Because it has uncertainty, \( \dot{V} \) is a differential inclusion:

\[ \dot{V} \in 2s_1(-Q_1 s_1 - P_1 sgn(s_1) + [\psi_1([x_e], [y_e], [\theta_e], [p]))
+ 2s_2(-Q_2 s_2 - P_2 sgn(s_2) + [\psi_2([x_e], [y_e], [\theta_e], [p]))], \]

with \([\psi_1([x_e], [y_e], [\theta_e], [p])\) and \([\psi_2([x_e], [y_e], [\theta_e], [p])\) bounded we can study the stability of the SMC and the influence of the tuning parameters on the robustness of the control loop.

For the first test case, the following uncertainties for the linear and angular velocities of the robot are considered:

\[ p_1 \in [-0.025,0.025] \]
\[ p_2 \in [-0.025,0.025] \]

The first set of SMC tuning parameters used in Section 3.3.2 were:

\[ k_0 = 10; \ k_1 = 3; \ k_2 = 7; \]
\[ P_1 = 0.1; \ Q_1 = 0.5; \]
\[ P_2 = 0.1; \ Q_2 = 0.5; \]

Using interval methods we are able to bound the uncertainty of the control law:

\[ \psi_1 \in [-0.175625,0.175625] \]
\[ \psi_2 \in [-0.425,0.4625] \]

Next, we can use SIVIA (Chapter 2, Section 2.3.6) to find the set defined by \( \dot{V}(s) < 0 \) and \( V(s) \geq 0 \), and the inputs to the algorithm can be defined as follows:

- the function: \( f(s) = 2s_1(-Q_1 s_1 - P_1 sgn(s_1) + \psi_1) + 2s_2(-Q_2 s_2 - P_2 sgn(s_2) + \psi_2); \)
- the variables: \( s = (s_1, s_2), s \in \mathbb{R}^2; \)
• the constraint: \( f(s) < 0 \Rightarrow Y = [\infty, 0] \). The solution set is \( S = f^{-1}([\infty, 0]) \);
• the precision: \( \varepsilon = 0.01 \).

The algorithm computes three sets:
• \( S_{\text{out}} \) – grey region in the plot;
• \( S_{\text{in}} \) – not present because there is no inner solution;
• \( S_{\text{boundary}} \) – yellow region in the plot;

![Figure 35. Sivia for \( \dot{V}(s) < 0 \)](image)

In Figure 35 the grey region represents the set \( \dot{V}(s) < 0 \), while the yellow region represents the uncertain set where \( \dot{V}(s) \) can be positive. In this case, the control law is \( V \)-stable for \( r = \max \left( [V](S_{\text{boundary}}) \right) = 0.73 \). It is obvious that this result is not satisfactory and the controller has to be retuned to increase the performance. This example illustrates how a small uncertainty in the measurements can have a great impact on overall performance due to the cumulative effect of all the parameters involved in the control law.

Next, the control law parameters are retuned while the surface parameters are kept the same:
\[
\begin{align*}
k_0 &= 10; \quad k_1 = 3; \quad k_2 = 7; \\
P_1 &= 0.4; \quad Q_1 = 1.5; \\
P_2 &= 0.4; \quad Q_2 = 1.5;
\end{align*}
\]

We use SIVIA again to find the set \( \dot{V}(s) < 0 \).
With the new set of parameters, the control law is V-stable for an arbitrary small \( r \) which depends only on the accuracy of the SIVIA algorithm.

With interval methods, the SMC parameters can be chosen in an informed manner. By increasing \( P_1, P_2 \) the controller becomes more robust, but these parameters cannot be increased without any consideration because they can add chattering which is not desirable.

### 3.5.2 Stability analysis of an SMC with uncertainty in the control output

For the SMC with the kinematic model without uncertainties we have:

\[
\dot{s} = \begin{pmatrix} \dot{s}_1 \\ \dot{s}_2 \end{pmatrix} = \begin{pmatrix} \dot{v}_r \cos \theta_e - v_r \dot{\theta}_e \sin \theta_e + \dot{y}_e \omega_d + \dot{y}_e \omega_d - \dot{v}_d + k_1 x_e \\ \dot{\omega}_r - \dot{\omega}_d + k_2 \dot{\theta}_e + k_0 \dot{y}_e \end{pmatrix} \tag{3.80}
\]

In this case, the controls are chosen as \( \dot{v}_r \) and \( \dot{\omega}_r \) and \( \dot{s} \), the control law with uncertainty, can be written as:

\[
\dot{s} = \begin{pmatrix} \dot{s}_1 \\ \dot{s}_2 \end{pmatrix} = \begin{pmatrix} (\dot{v}_r + p_1) \cos \theta_e - v_r \dot{\theta}_e \sin \theta_e + \dot{y}_e \omega_d + \dot{y}_e \omega_d - \dot{v}_d + k_1 \dot{x}_e \\ (\dot{\omega}_r + p_2) - \dot{\omega}_d + k_2 \dot{\theta}_e + k_0 \dot{y}_e \end{pmatrix}
\]

In a similar manner with Section 3.5.1 \( \dot{s} \) is a differential inclusion:

\[
\dot{s} \in -Q \cdot s - P \cdot \text{sgn}(s) + [\psi([x_e], [y_e], [\theta_e], [p])].
\]
where:

$$\psi = \begin{pmatrix} \psi_1 \\ \psi_2 \end{pmatrix} = \begin{pmatrix} p_1 \cos \theta_e \\ p_2 \end{pmatrix} \quad (3.81)$$

In this case $\psi_1$ and $\psi_2$ are easy to bound, but $p_1$ and $p_2$ belong to larger intervals which depend on the dynamics of the inner loop.

The uncertainty parameters for the control output and the SMC tuning parameters are:

- $p_1 \in [-1,1]$
- $p_2 \in [-1,1]$
- $k_0 = 10$; $k_1 = 3$; $k_2 = 7$
- $P_1 = 0.5$; $Q_1 = 1$
- $P_2 = 0.5$; $Q_2 = 1$

The bounds for the uncertainty of the control law are then computed:

- $\psi_1 \in [-1,1]$
- $\psi_2 \in [-1,1]$

Figure 37. Sivia for $\dot{V}(s) < 0$, with two different sets of parameters

For the second plot the control parameters were changed to:

- $P_1 = 1$; $Q_1 = 2$
- $P_2 = 1$; $Q_2 = 2$
3.5.3 Stability of SMC with uncertainty in the mathematical model of the system

An SMC using the dynamical model of the WMR was developed in Section 3.3.2. We can recall the dynamical model for the system without uncertainties:

\[
\begin{align*}
\ddot{x}_r &= \left(\frac{\tau_R}{R_R} + \frac{\tau_L}{R_L}\right) \cdot \cos \theta_r - \dot{y}_r \hat{\theta}_r \frac{m}{m} \\
\dot{y}_r &= \left(\frac{\tau_R}{R_R} + \frac{\tau_L}{R_L}\right) \cdot \sin \theta_r + \ddot{x}_r \hat{\theta}_r \\
\theta_r &= \left(\frac{\tau_R}{R_R} - \frac{\tau_L}{R_L}\right) \cdot L \cdot \frac{J}{J}
\end{align*}
\] (3.82)

We can have two sources of uncertainty present in the mathematical model of the system:

- parametric uncertainty due to the fact that the model is not accurate;
- parametric uncertainty due to faults.

In Section 3.3.2 the SMC was developed such that it could cope with faults in the wheels which lead to a variation in the radius:

\[
R_{Rreal} = R_{Rnom} + \Delta R_R \\
R_{Lreal} = R_{Lnom} + \Delta R_L
\] (3.83)

We can rewrite the mathematical model of the robot using intervals;

\[
\begin{align*}
\ddot{x}_r &= \left(\frac{\tau_R}{[R_R]} + \frac{\tau_L}{[R_L]}\right) \cdot \cos \theta_r - \dot{y}_r \hat{\theta}_r \frac{m}{m} \\
\dot{y}_r &= \left(\frac{\tau_R}{[R_R]} + \frac{\tau_L}{[R_L]}\right) \cdot \sin \theta_r + \ddot{x}_r \hat{\theta}_r \\
\theta_r &= \left(\frac{\tau_R}{[R_R]} - \frac{\tau_L}{[R_L]}\right) \cdot L \cdot \frac{J}{J}
\end{align*}
\] (3.84)

In this case, the fault can only make the radius of the wheel smaller, which means that \(R_R \in [R_{Rnom} - \Delta R_R, R_{Rnom}]\) and \(R_L \in [R_{Lnom} - \Delta R_L, R_{Lnom}]\).

For the SMC with the dynamical model with uncertainties we have:

\[
\begin{align*}
\dot{s} = \begin{pmatrix}
\dot{s}_1 \\
\dot{s}_2
\end{pmatrix} &= \begin{pmatrix}
\frac{\tau_R}{[R_R]} + \frac{\tau_L}{[R_L]} \cdot \frac{\cos \theta_e}{m} - \dot{x}_r \hat{\theta}_e \sin \theta_d - \dot{y}_r \hat{\theta}_e \cos \theta_d + \ddot{y}_e \omega_d + y_e \dot{\omega}_d - \nu_d + k_1 \dot{x}_e \\
\left(\frac{\tau_R}{[R_R]} - \frac{\tau_L}{[R_L]}\right) \cdot \frac{L}{J} \cdot \dot{\omega}_d + k_2 \hat{\theta}_e + k_0 \dot{y}_e
\end{pmatrix}
\end{align*}
\]
In this case, the uncertain terms cannot be clearly separated from the expression because they are at the denominator. But using interval methods, we are still able to bound the uncertainty in $\dot{s}$.

$$\psi = (\psi_1,\psi_2) = \left( \left( \frac{\tau_R}{[R_R]} + \frac{\tau_L}{[R_L]} \right) \cdot \frac{\cos \theta_e}{m} \right),$$

where

$$\frac{\tau_R}{[R_R]} - \frac{\tau_L}{[R_L]} \cdot \frac{L}{J}$$

The uncertain terms are bounded using the following set of parameters:

$$\tau_R, \tau_L \in [-3,3]$$  \hspace{1cm} (3.86)
$$R_R, R_L \in [0.08,0.1145]$$
$$\theta_e \in [-\pi, \pi]$$
$$m = 20$$
$$L = 0.245$$
$$J = 1.5$$

The differential inclusion control law $\dot{s}$ is computed as follows:

$$\dot{s} \in -Q \cdot s - P \cdot sgn(s)$$

$$+ \left( [\psi([\tau_R],[\tau_L],[\theta_e],[R_R],[R_L]) - [\psi([\tau_R],[\tau_L],[\theta_e],R_R,R_L)] \right)$$

After bounding the uncertainty we get:

$$\dot{s} \in -Q \cdot s - P \cdot sgn(s) + [0, 0.945] \times [0, 1.76]$$

The control parameters for the first plot in Figure 38 are:

$$k_0 = 10; \ k_1 = 3; \ k_2 = 7;$$
$$P_1 = 0.5; \ Q_1 = 1;$$
$$P_2 = 0.5; \ Q_2 = 1;$$

For the second plot in Figure 38 the parameters were changed to:

$$P_1 = 0.75; \ Q_1 = 2.5;$$
$$P_2 = 1.5; \ Q_2 = 4;$$
Figure 38. Sivia for $\dot{V}(s) < 0$, with two different sets of parameters

The result presented in Figure 38 proves the stability of the SMC which can tolerate the uncertainties listed in (3.86). Because the parametric uncertainty of the wheel radius was considered it means that the SMC can tolerate faults in both wheels (flat tyre).

3.6 Conclusions

This chapter explored multiple possibilities to improve the robustness and guarantee the stability of an SMC. Without reducing the generality of the proposed algorithms, the SMC was tested for a wheeled mobile robot.

In the first part of the chapter, a mathematical model of the robot was developed and validated. The model had to be complex enough to allow fault injection and replicate the robot behaviour in those circumstances. After the model was validated by comparing its behaviour with the real robot behaviour, it was used to test the algorithms proposed for SMC. This helped to speed up the theoretical development phase and, in some cases, to test situations that could not be implemented on the real robot (sudden flat tyre).

In the next part, a sliding mode controller based on the kinematic model was tested with and without the presence of a fault. When dealing with SMC for mobile robots, this type of controller is encountered most of the times in literature. From this starting point, we can compare the performance of the kinematic SMC with various types of controllers based on the dynamic model.

The first type of SMC based on the dynamic model is shown to have major problems when tested with the real system and the simulation. Firstly, the controller is only able to compensate for
small differences between the model and the real robot. Secondly, it was found that applying output torque control to the dc motors is not possible for the real robot. In most implementations, the torques are converted to velocities, which leads to having the dynamics of the robot handled by the inner PI controllers. Moreover, it is shown that this approach leads to identical behaviour between the two types of SMC: kinematic and dynamic.

To solve these problems, new control strategies found in the relevant literature are investigated. More robust control techniques for the dynamic model SMC are tested and implemented. They are shown to exhibit fault tolerance and robustness against parametric uncertainties. The second problem, applying torque control to the dc motors, is solved by developing a controller which uses current sensors for feedback.

A fault hiding approach which adds extra fault tolerant capabilities to the kinematic model SMC was proposed in Section 3.4. The kinematic model SMC, enhanced with the fault hiding block, is proved to be able to cope with faulty actuators. Moreover, using an SMC based on a reduced dynamic model for the mobile robot, it was possible to prove the stability of the kinematic SMC when the fault hiding block is included. This result allows for a simple fault hiding block to be added to the kinematic model SMC(used in the majority papers and applications of trajectory-tracking for mobile robots) to cope with faults in the actuators (flat tyre).

New techniques based on interval methods are employed in the last section to be able to find suitable tuning parameters for the SMC when dealing with multiple uncertainties. This result is extremely useful considering that there are no analytical methods for tuning an SMC. Another useful outcome for this method is the ability to estimate the stability and robustness margin in a guaranteed way. Multiple types of uncertainties were considered and interval methods proved to be a very powerful tool which was able to provide stable sets of parameters for all test cases.
Chapter 4: Stability analysis using energy functions with perturbations

4.1 Introduction

As it was mentioned in Chapter 2, in the 19th century, the Russian mathematician Aleksandr Mikhailovich Lyapunov introduced the most complete work regarding the stability analysis of dynamical systems [2]. His direct method is based on finding a function of the state coordinates of a dynamical system (Lyapunov function) which is positive-definite and its rate of change around the equilibrium point is always negative or zero.

Finding a Lyapunov function candidate is crucial for proving the stability of dynamical systems. While some more general methods have been developed for linear systems (Lyapunov indirect method) finding Lyapunov functions for most nonlinear systems is not a trivial matter. For mechanical systems, the energy function is a good candidate to be used as a Lyapunov function. Other approaches to construct Lyapunov function candidates include variable gradient, Krasovskii’s and Zuhov’s methods [50].

4.2 Illustrative example of stability analysis: the pendulum dynamical system

4.2.1 Stability analysis for the system without uncertainty

The model of a simple pendulum is:

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -\frac{G}{l} \sin(x_1) - bx_2
\end{align*}
\]

where:

- \(G\) - gravitational acceleration;
- \(l\) - pendulum length;
- \(m\) - pendulum mass;
- \(b\) - friction coefficient;

The energy function of the pendulum is a sum of its kinetic and potential energies:

\[
E(x_1, x_2) = \frac{1}{2} ml^2 x_2^2 + mGl(1 - \cos(x_1))
\]
We can choose a Lyapunov function candidate based on the scaled energy:

\[ V(x_1, x_2) = \frac{1}{2}x_2^2 + \frac{G}{l}(1 - \cos(x_1)) \]

The derivative for the Lyapunov function candidate is shown below:

\[ \dot{V}(x_1, x_2) = x_2\dot{x}_2 + \frac{G}{l}\sin(x_1)\dot{x}_1 = -\frac{G}{l}\sin(x_1)x_2 + \frac{G}{l}\sin(x_1)x_2 - bx_2^2 = -bx_2^2 \]

Because we want to use interval methods to test the Lyapunov function candidate, we define an energy level for which we can apply numerical methods as follows:

\[ g(x_1, x_2) = V(x_1, x_2) - 1 = \frac{1}{2}x_2^2 + \frac{G}{l}(1 - \cos(x_1)) - 1 \]

The pendulum system parameters used in the numerical examples are: \( \frac{G}{l} = 1, b = 0.65 \).

SIVIA algorithm (Section 2.3.6) is used to generate the pavings for the energy level function of the system. The inputs to the algorithm are defined as follows:

- the function: \( g(x) = \frac{1}{2}x_2^2 + \frac{G}{l}(1 - \cos(x_1)) - 1 \);
- the variables: \( x = (x_1, x_2), x \in \mathbb{R}^2 \);
- the constraint for the outer set: \( g(x) < 0 \Rightarrow \mathcal{Y} = [-\infty, 0] \). The solution set is \( \mathcal{X}_{out} = g^{-1}([-\infty, 0]) \);
- the constraint for the inner set: \( g(x) \geq 0 \Rightarrow \mathcal{Y} = [0, \infty] \). The solution set is \( \mathcal{X}_{in} = g^{-1}([0, \infty]) \);
- the precision: \( \varepsilon = 0.1 \).

The algorithm computes three sets (shown in Figure 40):

- \( \mathcal{X}_{out} \) – grey region in the plot;
- \( \mathcal{X}_{in} \) – white region in the plot;
- \( \mathcal{X}_{boundary} \) – yellow region in the plot, which contains parts of both outer and inner solution.
Figure 41 allows a more detailed inspection of how the energy function approximates the system’s trajectories. The cross-out condition \[ \frac{dg}{dx}(x) \cdot f(x), \] was evaluated in \( X_{\text{boundary}} \).

Figure 40. Paving for \( g(x_1, x_2) < 0 \). \( \dot{g}(x_1, x_2) \) is zero for \( x \in \{\pm \frac{\pi}{2}, 0\} \)

From the Lyapunov theory, we know that the system will converge to:

\[ \mathcal{A} = \{X| \dot{V}(X) = 0\} \]

In general, this set can be pessimistic, but in some cases, we can use La Salle theorem to find the largest positive invariant subset of \( \mathcal{A} \) towards which the system converges. For the pendulum we have the set \( \mathcal{R} = \{x_1 \in \mathbb{R}, x_2 = 0\} \) and the largest invariant set contained in \( \mathcal{R} \) is \( \mathcal{M} = (0,0) \). Using La Salle it is possible to prove that \( (x_1, x_2) = (0,0) \) is the only stable equilibrium point for the pendulum system.

In this chapter a numeric method is proposed to find Lyapunov functions that can work for a larger class of nonlinear dynamical systems. Using interval methods, we can solve the La Salle theorem numerically by finding positive invariant sets. To achieve this, we propose to add some small perturbation functions to the original energy level function chosen in such a way to eliminate all the points where \( \dot{g}(X) = 0 \).

A radial basis function is used as perturbation:

\[ P_i(X) = \varepsilon e^{-\frac{(X-c_i+\lambda f(c_i))^2}{2}} \]

where:

\[ (4.4) \]
- \( c_i \) are all the points where \( \dot{g}(X) = 0, \ i \in \{1, \ldots, N_c\} \) and \( N_c \) is the total number of points;
- \( \varepsilon, \lambda \) are parameters for the perturbation.

This type of perturbation function was chosen because its effects can be localised in small regions around the points where \( \dot{g}(X) = 0 \).

The energy level function with the added perturbation is:

\[
g_p(X) = g(X) + \sum_{i=1}^{N} P_i(X) = g(X) + \sum_{i=1}^{N} \varepsilon \cdot e^{-\left(X - c_i + \lambda f(c_i)\right)^2}
\]

(4.5)

The cross-out condition in one of the \( c_i \) points, for \( i = 1 \), can be derived as follows:

\[
 \frac{dg_p}{dX}(c_1) \cdot f(c_1) = \left( \frac{dg}{dX}(c_1) + \sum_{i=1}^{N} \varepsilon \frac{d}{dX} \left( e^{-\left(X - c_i + \lambda f(c_i)\right)^2}\right)(c_1) \right) \cdot f(c_1) =
\]

\[
= \frac{dg}{dX}(c_1) \cdot f(c_1) + \varepsilon \frac{d}{dX} \left( e^{-\left(X - c_i + \lambda f(c_i)\right)^2}\right)(c_1) \cdot f(c_1) + \sum_{i \neq 1} \varepsilon \frac{d}{dX} \left( e^{-\left(X - c_i + \lambda f(c_i)\right)^2}\right)(c_1) \cdot f(c_1) < 0
\]

For the pendulum dynamical system the perturbation functions will have the following expression:

\[
P_i(x_1, x_2) = \varepsilon \cdot e^{-\left[(x_1 - c_{i1} + \lambda f_1(c_{i1}))^2 + (x_2 - c_{i2} + \lambda f_2(c_{i2}))^2\right]}, \text{ where } c_1 = \left( +\frac{\pi}{2}, 0 \right), c_2 = \left( -\frac{\pi}{2}, 0 \right).
\]

Figure 42. Cross-out condition, \( \frac{dg_p}{dX}(x) \cdot f(x) \), evaluated in \%boundary (with perturbations \( \varepsilon = 0.1, \lambda = 0.5 \)).

Figure 43. The new \( g_p(x_1, x_2) = 0 \) with the perturbations added becomes an invariant set.
4.2.2 Stability analysis for the pendulum system with additive uncertainty

Because dynamical systems are always affected by uncertainty it is important to take this into account when we analyse their stability. The pendulum system will be used as a case study for stability analysis when we have to deal with additive uncertainty.

Pendulum model with additive uncertainty:

\[
\begin{align*}
\dot{x}_1 &= x_2 + [\alpha] \\
\dot{x}_2 &= -\frac{g}{l} \sin(x_1) - bx_2 + [\alpha]
\end{align*}
\]  

(4.6)

La Salle theorem cannot be used anymore to contract the set:

\[\mathcal{A} = \{X \mid \dot{V}(X) = 0\}\]

In this case, we have \(\dot{V}(x_1, x_2) > 0\) in the region around where \(x_2 = 0\) (\([\alpha] = [-0.1, 0.1]\)) which means that \(V(x_1, x_2)\) is no longer a Lyapunov function because the derivative \(\dot{V}\) is no longer negative semidefinite. For the system with additive uncertainty, there is no analytical method to guarantee the asymptotic stability of the system.

The energy level function candidate has the same analytical expression, but with bigger perturbations to counteract that \(\dot{V}(x_1, x_2) > 0\) in a region and not only a point:

\[\varepsilon = 0.52\]

\[\lambda = 1.3\]
Figure 46. The new $g_r(x_1, x_2) = 0$ with the perturbations added becomes a positive invariant set for the system with uncertainty.

4.3 Illustrative example of stability analysis: mass with damped spring system

4.3.1 Stability analysis for the system without uncertainty

The model of mass with damped spring system is [5]:

\[
\begin{cases}
    \dot{x}_1 = x_2 \\
    \dot{x}_2 = -\frac{b}{m} x_2 |x_2| - \frac{k_0}{m} x_1 - \frac{k_1}{m} x_1^3
\end{cases}
\]  

(4.7)

where:

- $m$ - mass;
- $b$ – damping coefficient;
- $k_0$ – spring constant;
- $k_1$ – nonlinear elasticity;

Energy function which is a sum of the kinetic and potential energies:

\[
V(x_1, x_2) = \frac{1}{2} m x_2^2 + \frac{1}{2} k_0 x_1^2 + \frac{1}{4} k_1 x_1^4
\]  

(4.8)

In order to find a positive invariant set for this system, we consider an energy level curve as candidate:

\[
g(x_1, x_2) = V(x_1, x_2) - 1 = \frac{1}{2} m x_2^2 + \frac{1}{2} k_0 x_1^2 + \frac{1}{4} k_1 x_1^4 - 1
\]  

(4.9)
For numerical illustration, the system parameters are $m = 1$, $b = 1$, $k_0 = 1$ and $k_1 = 5$. In a similar manner with the pendulum system the perturbations are applied in two points.

**Figure 47.** Paving for $g(x_1, x_2) < 0$. $\dot{g}(x_1, x_2)$ is zero for $x \in \{ \pm 0.8464, 0 \}$.

**Figure 48.** Cross-out condition evaluated when $g(x_1, x_2) = 0$.

**Figure 49.** $\dot{g}_P(x_1, x_2)$ is negative on the border and $g_P(x_1, x_2) = 0$ becomes an invariant set.

**Figure 50.** Cross-out condition evaluated when $g_P(x_1, x_2) = 0$ with perturbation ($\varepsilon = 0.1, \lambda = 0.4$).
4.3.2 Stability analysis for the system with additive uncertainty

Similarly with the pendulum system, the mass with damped spring system will be analysed when the model has additive uncertainty.

System equations with additive uncertainty are shown below:

\[
\begin{aligned}
\dot{x}_1 &= x_2 + [\alpha] \\
\dot{x}_2 &= -\frac{b}{m}x_2|_{x_2} - \frac{k_0}{m}x_1 - \frac{k_1}{m}x_1^3 + [\alpha]
\end{aligned}
\]  

(4.10)

Because of the uncertainty \( ([\alpha] = [-0.1, 0.1]) \), the energy function derivative \( \dot{g}(x_1, x_2) \) becomes positive around \( x_2 = 0 \).

\[\text{Figure 51. Cross-out condition for } g(x_1, x_2) = 0, \quad \dot{g}(x_1, x_2) \text{ is positive in two regions.}\]

\[\text{Figure 52. Cross-out condition for } g_p(x_1, x_2) = 0 \text{ with perturbations } (\varepsilon = 0.52, \lambda = 0.445).\]

\[\text{Figure 53. } \dot{g}_p(x_1, x_2) \text{ is negative on the border and } g_p(x_1, x_2) < 0 \text{ becomes a positive invariant set.}\]

107
The energy level function candidate had the same analytical expression, but with bigger perturbations to counteract that $\dot{V}$ was positive in a region and not only a point.

### 4.4 Illustrative example of stability analysis: limit cycle system

#### 4.4.1 Stability analysis for the system without uncertainty

The dynamical system is defined by the following equations [5]:

$$
\begin{align*}
\dot{x}_1 &= 4x_1^2 x_2 - x_1 (x_1^2 + 2x_2^2 - 4) \\
\dot{x}_2 &= -2x_1^3 - x_2 (x_1^2 + 2x_2^2 - 4)
\end{align*}
$$

The limit cycle for this system is defined by:

$$
\{(x_1, x_2) \in \mathbb{R} \times \mathbb{R}: x_1^2 + 2x_2^2 - 4 = 0\}
$$

We chose a Lyapunov function:

$$
V(x_1, x_2) = (x_1^2 + 2x_2^2 - 4)^2,
$$

$$
\dot{V}(x_1, x_2) = 2(x_1^2 + 2x_2^2 - 4) \frac{d}{dt} (x_1^2 + 2x_2^2 - 4) = -2(2x_1^2 + 2x_2^2 - 4)^2 (2x_1^2 + 4x_2^2).
$$

We have $\dot{V}(x_1, x_2) < 0$ for $(x_1, x_2) \in \mathbb{R} \times \mathbb{R}$ and $(x_1, x_2) \neq (0,0)$. The origin $(0,0)$ is a local maximum and it is an unstable point. It can be analytically proven that the limit cycle is attractive [5].

Define $D_c = \{(x_1, x_2) \in \mathbb{R} \times \mathbb{R}, V(x_1, x_2) < \beta\}$ we can test the cross-out condition $(V(\dot{X}) = \frac{dV}{dX}(X) \cdot f(X)$ and check that it is negative on all the border regions for any arbitrary chosen $\beta$. In this case, we have proven $D_c$ to be an invariant set using interval methods.

![Figure 54](image.png)

*Figure 54. Paving for $V(x_1, x_2) < 1$. The cross-out condition $\frac{dV}{dX}(X) \cdot f(X)$ is negative on the border.*

108
4.4.2 Stability analysis for the system with uncertainty

System equations with additive uncertainty:

\[
\begin{align*}
\dot{x}_1 &= 4x_1^2x_2 - x_1(x_1^2 + 2x_2^2 - 4) + [\alpha] \\
\dot{x}_2 &= -2x_1^3 - x_2(x_1^2 + 2x_2^2 - 4) + [\alpha]
\end{align*}
\] (4.13)

\[
\dot{V}(x_1, x_2) = 2(x_1^2 + 2x_2^2 - 4) \frac{d}{dt}(x_1^2 + 2x_2^2 - 4) = 2(x_1^2 + 2x_2^2 - 4)(2x_1\dot{x}_1 + 4x_2\dot{x}_2) = \\
= 2(x_1^2 + 2x_2^2 - 4) [(x_1^2 + 2x_2^2 - 4)(-2x_1^2 - 4x_2^2) + 2x_1[\alpha] + 4x_2[\alpha]]
\]

In this case, the system trajectories will not converge to the limit cycle anymore, because \(\dot{V}(x_1, x_2)\) becomes positive on a region around the limit cycle defined above in Figure 55. Moreover, the unstable point in (0,0) also becomes a region. Hence, there is no analytical method in the literature to prove the stability of the dynamic system with uncertainties (4.13). In this thesis, a new method based on interval analysis is proposed to find the smallest invariant set for such dynamic systems with uncertainties.

After we find the set \(\mathcal{V}_{dp} = \{(x_1, x_2) \in \mathbb{R} \times \mathbb{R} \mid \dot{V}(x_1, x_2) \geq 0\}\) (Figure 56) the smallest energy level \(\beta\) for which \(V(x_1, x_2) < \beta\) is an invariant set can be estimated. For the uncertainty level \([\alpha] = [-0.1, 0.1]\) we find \(\beta = 0.22\) (Figure 57). In order to prove that all system trajectories converge towards \(V(x_1, x_2) < 0.22\) and not towards the region around (0,0) we have to find the biggest

\[\text{Figure 55. Plot for the set } \dot{V}(x_1, x_2) \geq 0.\]

\[\text{Figure 56. Plot for } \dot{V}(x_1, x_2) \text{ on the border of } V(x_1, x_2) < 0.1. \text{ } \dot{V}(x_1, x_2) \text{ is positive, hence } V(x_1, x_2) < 0.1 \text{ is not an invariant set.}\]
invariant set that does not include that region. We already found the set for $\dot{V}(x_1, x_2) \geq 0$ which contains two distinct regions, one around zero and one around the limit cycle. If we can prove that no system trajectory converges towards one of the two regions, it means that the region is a local maximum and the other is the local minimum. Figure 58 shows the largest invariant set found for the system with uncertainty, which has the energy function given by $V(x_1, x_2) < 15.61$. This proves that the region around (0,0) is not a stable equilibrium point and is a local maximum.

$$\text{Figure 57. } V(x_1, x_2) < 0.22 \text{ is the smallest invariant set for the system with uncertainty.}$$

$$\text{Figure 58. Plot for the biggest invariant set that does not include (0,0). This was achieved for } V(x_1, x_2) < 15.61.$$

4.5 Finding the perturbation parameters using contractors

4.5.1 Perturbation parameter contractor definition

As explained in the previous chapters, we are trying to find a positive invariant set for a system by adding some small perturbations to the existing energy function. The perturbation function is shown below:

$$P_i(X) = \epsilon e^{-(X - c_i + \lambda f(c_i))^2} \tag{4.14}$$

The new boundary function for the set is:

$$g_P(X) = g(X) + \sum_{i=1}^{n} P_i(X) = g(X) + \sum_{i=1}^{n} \epsilon e^{-(X - c_i + \lambda f(c_i))^2} \tag{4.15}$$
In the examples addressed in the previous chapters, we have \( \dot{g}(x_1, x_2) = 0 \) for some points and \( \dot{g}(x_1, x_2) < 0 \) for the remaining state space (\( g \) is chosen as energy function and the system is without uncertainty). To solve this problem, we can choose the perturbation parameters \((\epsilon, \lambda)\) as small as possible and we can get \( g_p(x_1, x_2) \) as a boundary for a positive invariant set.

However, if we have uncertainty in the system, choosing the parameters for the perturbation is not straightforward anymore because \( \dot{g}(x_1, x_2) \) is positive in some regions. This means that \( g(x_1, x_2) \) is no longer a Lyapunov function. However, we will consider \( g(x_1, x_2) \) as a Lyapunov function candidate. Hence, bigger values have to be chosen for the perturbation parameters, which means that the overall effect on the resulting function will be unpredictable. This happens because the derivative of the Lyapunov function candidate is affected by these perturbations not only around the perturbated points. Hence, in some other points the derivative of the Lyapunov function candidate can become positive. We propose in the following, a method to search the parameters of the perturbation function which guarantees the negativeness of the derivative of the Lyapunov function.

The feasible solution set for \((\epsilon, \lambda)\) has to satisfy the following system:

\[
\begin{align*}
\{ & g_p(x_1, x_2, \epsilon, \lambda) = 0 \\
& \dot{g}_p(x_1, x_2, \epsilon, \lambda) < 0 \}
\end{align*}
\]  

(4.16)

This system can be solved using SIVIA and contractors in order to find inner and outer approximations.

4.5.2 Perturbation parameter estimation for the pendulum system

Figure 59 shows a paving of the feasible solution set for \((\epsilon, \lambda)\) for the system without uncertainty. Any set of values chosen from the white region (inner approximation) for \((\epsilon, \lambda)\) will guarantee that \( g_p(x_1, x_2, \epsilon, \lambda) = 0 \) is an invariant set. Boxes in the yellow region belong to the border of the solution set and they might have part of the solution. The size of the yellow region depends on the chosen accuracy of the SIVIA algorithm when estimating \( g_p(x_1, x_2, \epsilon, \lambda) = 0 \).

Figure 60 shows a paving of the solution set with better accuracy. The inner approximation (white) is bigger and the border (yellow) is smaller. The higher accuracy increases the computation time, which means the algorithm should be focused on a smaller search space.
Figure 59. Paving of the feasible solution set for the system without uncertainty.

Figure 60. Paving of the feasible solution set with better accuracy.

Figure 61 and Figure 62 show a paving for the \((\varepsilon, \lambda)\) solution set when the system has additive uncertainty \([\varepsilon] = [-0.05,0.05]\) and \([\varepsilon] = [-0.1,0.1]\). The inner approximations are smaller compared to the previous case (system without uncertainty).

Figure 61. Paving of the feasible solution set for the system with additive uncertainty \([\varepsilon] = [-0.05,0.05]\).

Figure 62. Paving of the feasible solution set for the system with additive uncertainty \([\varepsilon] = [-0.1,0.1]\).

From Figure 61, we notice that the inner approximation is smaller, close to the point when it vanishes. The perturbation cannot compensate for bigger uncertainty in the system equations. We might need to choose a different function for the perturbation to overcome bigger uncertainties, but in some cases it is possible that no positive invariant set exists for the system.
Figure 63 shows a paving for \( g_p(x_1, x_2, \varepsilon, \lambda) = 0 \) with \( \varepsilon = [0, 2.5] \) and \( \lambda = [0, 2.5] \). Every yellow box from the border region can be used to define a constraint for finding the feasible set \((\varepsilon, \lambda)\) for which \( \dot{g}_p(x_1, x_2, \varepsilon, \lambda) < 0 \).

4.5.3 Perturbation parameter estimation for the mass with damped spring system

For system (4.7), the contractor for the feasible solution set behaves in a similar manner as with the pendulum system. The feasible solution set decreases as the additive uncertainty in the system model increases. The main difference is that this system is more sensitive to variations in the perturbation parameters. Choosing the optimal set of \((\varepsilon, \lambda)\) parameters is not straightforward without applying the contractors.
4.5.4 Finding positive invariant sets for systems with unknown energy function

In some cases, when the energy function of the dynamic system is unknown, which is the case for non-mechanical systems, we can still use the techniques presented above to find a positive invariant set for the system. We can replace the energy function used in the previous sections with another basic function to intend to approximate the system behaviour. However, because this function is not a Lyapunov candidates, we have to use perturbation functions to adjust it in order to have the cross-out condition negative in all points on the border of the invariant set.

In the general form of the energy function with added perturbations, we can replace $g$ with any function $g_F(\mathbf{X})$ that can approximate the system trajectories:

$$
g_F(\mathbf{X}) = g_F(\mathbf{X}) + \sum_{i=1}^{n} P_i(\mathbf{X}) = g_F(\mathbf{X}) + \sum_{i=1}^{n} \varepsilon e^{-((X-e_i+\lambda f(c_i))^2)}
$$

For the pendulum system we propose to approximate the energy levels of the system with a circle:

$$
g_F(x_1, x_2) = x_1^2 + x_2^2 - 1
$$

Next, we have to test where the cross-out condition $\frac{dg_F}{d\mathbf{X}} \cdot f(\mathbf{X})$ is positive. Two points are found which means we have to add at least two perturbation functions.

Figure 68 shows a paving for the solution set for $(\varepsilon, \lambda)$ when the pendulum system energy function is approximated with a circle. After we chose a set of parameters for the perturbations from the inner approximation, the new function defines a positive invariant set for the system (Figure 69).
Figure 68. Paving of the feasible solution set for $(\varepsilon, \lambda)$ when the system energy is approximated with a circle.

For the mass with a damped spring system, the energy levels are approximated better with an ellipse:

$$g_F(x_1, x_2) = \frac{x_1^2}{0.35} + \frac{x_2^2}{1} - 1$$  \hfill (4.19)

Figure 70 shows a paving for the solution set for $(\varepsilon, \lambda)$ when the energy function is approximated with an ellipse. Figure 71 shows the positive invariant for the system after a set of parameters is chosen from the white region (inner approximation).

Figure 69. Positive invariant set obtained with perturbation parameters: $\varepsilon = 0.5, \lambda = 1$.

Figure 70. Paving of the feasible solution set for $(\varepsilon, \lambda)$ when the system energy is approximated with an ellipse.

Figure 71. Positive invariant set obtained with perturbation parameters: $\varepsilon = 0.36, \lambda = 67$. 
4.6 Conclusions and future work

4.6.1 Conclusions

The problem of finding positive invariant sets using interval methods was addressed in this chapter. This is an Eulerian approach to study the stability of nonlinear dynamical systems. Because there is a lack of methods in the literature for stability analysis of uncertain dynamical systems from this Eulerian perspective, interval analysis techniques were used to address it.

Based on LaSalle’s invariance principle a new automatic method for finding positive invariant sets for nonlinear dynamical systems is proposed. It is shown that by adding small perturbations to the energy function of nonlinear mechanical systems we can eliminate the problematic points were the Lyapunov function derivative is negative semi-definite.

Lyapunov stability theory can be successfully applied to some nonlinear systems, but it cannot deal with systems uncertainties. Interval methods are used to study the stability of nonlinear systems with additive uncertainty and even in the presence of significant uncertainties.

In the last part, a contractor method with SIVIA algorithm is used to estimate a guaranteed set of parameters for the perturbation functions. This method can also be used for a more general class of nonlinear systems when an energy function is not available.
4.6.2 Future work

Future work plans include the extension of the work conducted in this chapter for autonomous nonlinear systems to time-variant systems. Stability analysis for this class of systems is difficult because it is not easy to find Lyapunov functions for them. Barbalat’s lemma has been used to prove the stability for such systems, but it needs some assumptions about the bounds of the time-variant system.

Some preliminary work includes the analysis of the time-variant pendulum system which is illustrated in the figures below.

Figure 72. Lyapunov function plot for the time-variant pendulum system \((t=0.57)\).

Figure 73. Lyapunov function plot for the time variant pendulum system \((t=0,6.28))\)
Chapter 5: Guaranteed integration of ODEs with uncertainties

5.1 Introduction

This chapter proposes a new method for dynamically bounding (bracketing) the manifold containing the solution of a nonlinear ODE. The computation of invariant sets is widely used in the control theory for proving stability [85]. For linear systems, there are already well-established methods to determine invariant sets. However, for nonlinear systems described by ODEs, the systematic computation of invariant manifolds is still an open research problem and there are less general approaches and mostly developed in discrete-time [86].

The key idea is to iteratively generate parametrised tubes [21] [87] that contain the upper and lower bounds [88] of the invariant manifold. These tubes are generated starting from a seed set that is propagated iteratively using a combination of a state propagation [44], cross-out condition [22] and Picard theorem [89]. A measure of overestimation of the upper and lower bounds that allows controlling integration time is proposed. This measure also allows characterizing the family of ODEs that avoids the explosion of uncertainty in the state propagation. An algorithmic implementation is also proposed.

The structure of the chapter is as in the following: Section 5.2 summarises the necessary background for developing the proposed approach. In Sections 5.2.2 and 5.2.3, two methods to approximate an invariant manifold are presented. The first method is based on Picard and Brouwer theorems [21] (a Lagrangian method), and the second method is based on capture tubes [21] (an Eulerian method). Section 5.3 and 5.4 propose two new methods to approximate an invariant manifold, one based on RBFNs, and the second one based on linear interpolation. An algorithmic implementation and a case study based on an existing example proposed in the literature will be used for illustrating the methods proposed in Sections 5.3 and 5.4.
5.2 ODE integration using interval methods

5.2.1 Guaranteed integration – a Lagrangian approach

Guaranteed integration is a set of techniques, which allow to compute a tube that encloses the solution of a state equation or to enclose all solutions of a differential inclusion. It uses Picard-Lindelöf operator and the Brouwer theorem [21].

**Brouwer theorem.** Any continuous function $f(x)$ mapping a compact convex set $X$ to itself has a fixed point:

$$\exists x \in X \mid f(x) = x$$

Example proposed by Jaulin [21]: Consider $f(x) = \sin(x) \cdot \cos(x)$ and $X = [-2,2]$. Since:

$$f([-2,2]) \subseteq \sin([-2,2]) \cdot \cos([-2,2]) = [-1,1] \subset X$$

Then, applying the Brouwer theorem, we have:

$$\exists x \in [-2,2] \mid \sin(x) \cdot \cos(x) = x$$

Consider a nonlinear system $\dot{x} = f(x)$, where $f(x)$ is a Lipschitz continuous function, and its known initial condition $x_0^*$. The Picard-Lindelöf operator is able to calculate the interval enclosure for the trajectory $x^*(t)$ and it is defined as:

$$\mathcal{T}: x(t) \rightarrow t \rightarrow x_0^* + \int_0^t f(x(\tau))d\tau$$

where $\mathcal{T}$ is the tube which encloses the solution.

Because $f$ is Lipschitz continuous, the Picard-Lindelöf operator has a unique fixed point correspondent to the solution $x^*(t)$. For an interval tube $[x](t)$, for $\forall t$, $[x](t)$ is a box of $\mathbb{R}^n$. From the Brouwer theorem:

$$\mathcal{T}([x](t)) \subseteq [x](t) \Rightarrow x^*(t) \subset [x](t)$$

The representation of the tube $[x](t)$ together with its correspondent Picard-Lindelöf operator $\mathcal{T}([x](t))$ is illustrated in Figure 74. It can be noticed that $\mathcal{T}([x](t)) \subset [x](t)$ is valid only for $t \in [0, t_1]$. If we apply this restriction to the tube $\mathcal{T}$, we get the inclusion. Hence

$$\forall t \in [0, t_1], x^*(t) \in \mathcal{T}([x](t))$$

where $t_1$ can be defined as:

$$t_1 = \sup\{t \in \mathbb{R}^+ | \forall t \in [0, t], \mathcal{T}([x](t)) \subset [x](t)\}$$
5.2.2 Positive invariant sets – an Eulerian approach

The idea of positive invariance can be easily understood by referring to a simple autonomous system in state space form:

$$\dot{x} = f(x)$$  \hspace{1cm} (5.1)

It is assumed that the above system of equations is defined in a proper open set $\mathcal{O} \subset \mathbb{R}^n$ and that there exists a globally defined solution (i.e., for all $t \geq 0$) for every initial condition $x(0) \in \mathcal{O}$. Although the concept has been already considered, positive invariance is formally defined in the following definition.

**Definition 5.1.** (Positive invariance). The set $S \subseteq \mathcal{O}$ is said to be positively invariant w.r.t. (eq 5.1) if every solution of equation 5.1 with initial condition $x(0) \in S$ is globally defined and such that $x(t) \in S$ for $t > 0$.

According to [85], Nagumo’s theorem is of fundamental importance in the characterization of positively invariant sets for continuous-time systems. The best way to state the theorem is to consider the notion of tangent cone.

**Theorem 5.1.** (Nagumo, 1942) [85]. Consider the system $\dot{x} = f(x(t))$ and assume that for each initial condition $x(0)$ in an open set $\mathcal{O}$ it admits a (not necessarily unique) solution defined for all $t \geq 0$. Let
𝕊 ⊂ ℤ be a closed set. Then, ℤ is weakly positively invariant for the system if and only if the velocity vector satisfies Nagumo’s condition:

\[ f(x) \in T_S(x), \text{ for all } x \in ℤ \]  \hspace{1cm} (5.2)

The theorem has a simple geometric interpretation. Indeed, in plain words, it says that if the velocity vector \( \dot{x} = f(x(t)) \), “points inside or it is tangent to \( ℤ \)”, for all \( x \) on the border of \( ℤ \), i.e., \( x \in \partial ℤ \), then the trajectory \( x(t) \) remains in \( ℤ \) for all \( t \).

From the practical point of view, the tangent cone is given by:

\[ T_S(x) = \{ z | \nabla g_i(x)^T z \leq 0, \text{ for all } i \in Act(x) \} \]  \hspace{1cm} (5.3)

where \( Act(x) \) denotes the set of active constraints \( Act(x) = \{ i | g_i(x) = 0 \} \).

5.2.2.1 Capture tubes as positive invariant sets

Consider a time-invariant nonlinear dynamical system:

\[ S_f: \dot{x}(t) = f(x(t)) \]  \hspace{1cm} (5.4)

and an initial condition \( x(0) \in D_0 \).

- \( f: D \to \mathbb{R}^n \) is continuous in \( x \);
- \( D \) is an open subset of \( \mathbb{R}^n \) with \( 0 \in D \);
- \( D_0 \) is a closed subset of \( \mathbb{R}^n \) (it can be an arbitrary set or a box \( [x_0] \in \mathbb{R}^n \)).

We want to find the solution set for the evolution of the dynamical system for \( t \in [0,t_f] \).

In order to accomplish the above task, we have to define the capture tube for all the states starting from the initial set \( D_0 \). A tube \( G \) is an interval of trajectories and it can be represented using inequalities:

\[ G(t) = \{ x, g(x,t) \leq 0 \} \]  \hspace{1cm} (5.5)

A tube is said to be a capture tube if the fact that \( x(t) \in G(t) \) implies that \( x(t + t_1) \in G(t + t_1) \) for all \( t_1 > 0 \).

The following theorem [22] will be used to find an arbitrary small capture tube for the solution set of our dynamical system.
Theorem 5.2. (Capture tubes) [22]. Consider a tube \( \mathcal{G}(t) = \{ x, g(x, t) \leq 0 \} \) where \( g: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m \). If the cross out condition:

\[
\begin{align*}
(i) & \quad \frac{\partial g_i}{\partial x}(x, t) \cdot f(x) + \frac{\partial g_i}{\partial t}(x, t) \geq 0 \\
(ii) & \quad g_i(x, t) = 0 \\
(iii) & \quad g(x, t) \leq 0
\end{align*}
\] (5.6)

is inconsistent for all \( x \), all \( t > 0 \), and all \( i \in \{ 1, \ldots, m \} \) then \( \mathcal{G}(t) \) is a capture tube for the time dependent system \( \dot{x} = f(x, t) \).

Because the dynamical system is time invariant, the capture tube will also be defined as time invariant:

\[ \mathcal{G}: \{ x, g(x) \leq 0 \} \] (5.7)

As it can be seen from the above theorem, in order to find the capture tube for the solution set of the dynamical system starting from \( \mathcal{D}_0 \) we have to find a function \( g(x) \) for which the cross out condition is inconsistent. Hence the Eulerian method for stability.

5.3 ODE solution boundary approximation using radial basis functions

A radial basis function network (RBFN) can be classified as being part of artificial neural networks. The output of an RBFN is a linear combination of radial basis functions of its inputs. RBFNs have many applications, and they have been used for: approximating nonlinear functions, classification, feature recognition, system identification and control etc.

Radial basis functions can interpolate multivariate nonlinear functions from a given set of values. They have been used to solve differential equations [90] [91] and also to construct Lyapunov functions to analyse the stability of nonlinear dynamical systems [92].

There exist many numerical methods for solving differential equations, but the majority of them cannot deal with an uncertain initial condition or uncertain parameters. Interval analysis is a mathematical tool which can deal with uncertainty in a guaranteed way. A combination of existing methods for solving differential equations with interval methods could lead to a guaranteed approach. The overestimation, one of the biggest problems when using interval methods, will be minimised using the proposed algorithm. In the next part, a combination of radial basis functions with interval methods will be used to bracket the solution of an ordinary differential equation with uncertain initial condition.
5.3.1 Radial basis function network - general aspects

5.3.1.1 Network structure

A typical RBFN has three layers:

- an input layer with network inputs as vectors of real numbers \( \mathbf{x} \in \mathbb{R}^m \);
- a hidden layer with the RBF activation function;
- an output layer with the network output.

The output of the RBFN, \( \psi: \mathbb{R}^m \rightarrow \mathbb{R} \), is a linear combination of basis functions:

\[
y = \psi(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i \phi_i(\|\mathbf{x} - \mathbf{c}_i\|)
\]  

(5.8)

The basis functions can have many forms, but the most commonly used is the Gaussian with Euclidian norm:

\[
\phi_i(\mathbf{x}) = \exp\left(-\frac{(\mathbf{x} - \mathbf{c}_i)^2}{r^2_i}\right)
\]  

(5.9)

where:

- \( n \) is the number of neurons in the hidden layer \( i \in \{1, 2, \cdots, n\} \);
- \( \phi_i \) is the radial basis activation function of the \( i^{th} \) neuron;
- \( w_i \) is the weight of the \( i^{th} \) neuron;
- \( w_0 \) is the network output bias;
- \( c_i \) is the center of the \( i^{th} \) neuron;
- \( r_i \) is the width of the \( i^{th} \) neuron;
5.3.1.2 Training methods

RBFNs are trained using pairs of input-target datasets \( \{x_j, y_j\} \) with \( j \in \{1, 2, \cdots, N\} \), where \( N \) represents the size of the training set. Several training algorithms can be used to adjust network parameters: the centers \( c_i \), the widths \( r_i \) and finally the weights \( w_i \). In a typical approach, the centers and widths of the basis functions are adjusted in the first step and then the weights in the second step.

One of the most common training methods used is based on gradient descent optimisation techniques. Least squares optimisation criteria is usually preferred:

\[
E(w, c, r) = (y_j - \psi(w, c, r))^2
\]  

Updating the network parameters for each \( y_j \) element in the training data set is done as follows:

\[
w_i = w_i - \eta \frac{\partial E}{\partial w_i} = w_i - 2\eta \phi_i (y_j - \psi_j(w, c, r))
\]  

\[
c_i = c_i - \eta \frac{\partial E}{\partial c_i} = c_i - 2\eta w_i \frac{\partial \phi_i}{\partial c_i} (y_j - \psi_j(w, c, r))
\]  

\[
r_i = r_i - \eta \frac{\partial E}{\partial r_i} = r_i - 2\eta w_i \frac{\partial \phi_i}{\partial r_i} (y_j - \psi_j(w, c, r))
\]  

where \( \eta \) is the learning rate.

When interpolating a function \( f: \mathbb{R}^m \rightarrow \mathbb{R} \) from a number of known points \( y_j = f(x_j) \) the centers \( c_i \) can be chosen as \( x_j \) with \( i, j \in \{1, 2, \cdots, N\} \). In this case, finding the weights of the network amounts to solving a linear system of equations:

\[
\begin{pmatrix}
g_{11} & g_{12} & \cdots & g_{1N} 
g_{21} & g_{22} & \cdots & g_{2N} 
\vdots & \vdots & \ddots & \vdots 
g_{N1} & g_{N2} & \cdots & g_{NN}
\end{pmatrix}
\begin{pmatrix}
w_1 
w_2 
\vdots 
w_N
\end{pmatrix} =
\begin{pmatrix}
y_1 
y_2 
\vdots 
y_N
\end{pmatrix}
\]  

where \( g_{ij} = \phi_i(\|x_j - x_i\|) \).

In matrix form we can write:

\[
G \cdot w = y
\]  

\[
w = G^{-1} \cdot y
\]  

where \( w \) contains the trained weights of the RBFN.
Example: Approximating a sine function using an RBFN. The network has a size of 15 radial basis functions and it was trained for 100 steps. The input data set consists of 100 values obtained from the sine function. The following set of plots shows the evolution of error (Figure 77) and network weights (Figure 76), while the last plot (Figure 78) shows the training result and each radial basis function.

Figure 76. Weights evolution during training.

Figure 77. Output error evolution during training.

Figure 78. Sine function approximated with an RBFN with 15 basis functions.
An algorithm summarising the training process is shown in the following table:

**Table 5.1.**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RBF TRAIN (in: ({X, Y}, n_w); out: (\psi))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>(w_i = \text{InitRandom}(), i \in {1, 2, \ldots, n_w});</td>
</tr>
<tr>
<td>2. do</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>(w_i = w_i - 2\eta \phi_i (y_j - \psi_j(x_j, w))), (i \in {1, 2, \ldots, n_w}, x_j \in X, y_j \in Y);</td>
</tr>
<tr>
<td>4. while</td>
<td>stop_condition == false</td>
</tr>
<tr>
<td>5. return</td>
<td>(\psi);</td>
</tr>
</tbody>
</table>

For the RBFN presented above only the weights were trained. The *stop condition* can take multiple forms:

- stop after a fixed number of steps;
- stop if the error is smaller than a predefined threshold;
- stop if the error gradient is smaller than a predefined threshold;

5.3.2 Radial basis function networks with interval methods

In this paragraph, RBFN will be used to find the function which approximates the capture tube, \(g(x)\). Recall that the RBFN has the output function (5.8) which implies that one trivial approach will be to chose \(g(x) = \psi(x)\). The RBFN will have to be trained such that:

\[
\begin{align*}
\psi(x) &= 0, \text{on the boundary of the capture tube} \\
\psi(x) &< 0, \text{inside the capture tube} \\
\psi(x) &> 0, \text{outside the capture tube}
\end{align*}
\]  

(5.14)

Because in our case the problem is to find only the boundary of the capture tube we can take advantage of this requirement and reduce the complexity of the problem with one dimension. We define a new state vector \(\zeta \in \mathbb{R}^{n-1}\) such that:

\[
\zeta = \{x_1, x_2, \ldots, x_n\} \setminus \{x_p\} = \{x_1, \ldots, x_{p-1}, x_{p+1}, \ldots, x_n\}
\]  

(5.15)

The dimension for the variable \(x_p\) is chosen based on the distribution of the training data set (on the boundary of the capture tube). Then the capture tube function is chosen as follows:

\[
g(x) = \psi(\zeta) - x_p
\]  

(5.16)

which means that the RBFN will have to be trained such that:
\[
\begin{align*}
\begin{cases}
g(x) = 0 \\
\frac{\partial g}{\partial x}(x).f(x) < 0 \Rightarrow \frac{\partial \psi}{\partial \zeta}(\zeta).f_\zeta(x) - f_p(x) < 0
\end{cases}
\end{align*}
\] (5.17)

In this approach, we have to train the RBFN to approximate the function not only by its values but also by its derivative values. RBFN have been used successfully to approximate solutions for higher order differential equations [90] and new, more advanced, methods have been proposed since [91] [93]. A very similar approach with the one we study is presented in [92] where RBFN is used to find a Lyapunov function candidate to study the stability of nonlinear dynamical systems. In [94] [95] [96] it is shown that the radial basis function approach can be generalised to approximate a function not by its direct values, but by the values of a linear operator applied to the function.

Moreover, in [92] it is shown that for any dynamical system there exists a Lyapunov function \( V \) such that:

\[
\dot{V}(x) = -\bar{c}, \text{ where } \bar{c} > 0
\] (5.18)

By deriving \( \dot{V} \) we obtain a first order partial differential equation:

\[
\dot{V}(x) = \frac{\partial V}{\partial x}(x).f(x) = -\bar{c}
\] (5.19)

The approximation of the solution of this equation via RBFN is a Lyapunov function. Similarly, we can apply the same principle to find an approximation for \( g(x) \), the function which characterizes the boundary of the capture tube:

\[
\frac{\partial g}{\partial x}(x).f(x) = -\bar{c}
\] (5.20)

The main difference is that in our case the approximated function is defined only on the boundary of the capture tube and not on the entire state space of the dynamical system.

As shown in 5.17, we have:

\[
\frac{\partial g}{\partial x}(x).f(x) = \frac{\partial \psi}{\partial \zeta}(\zeta).f_\zeta(x) - f_p(x) = -\bar{c}
\] (5.21)

\[
\frac{\partial \psi}{\partial x}(x).f(x) = \sum_{i=1}^{n} w_i \frac{\partial \phi_i}{\partial x}(||x - c_i||) f(x)
\]
Because we have a Gaussian basis function:

$$\phi_i(x) = \exp\left(-\frac{(x - c_i)^2}{r_i^2}\right)$$  \hspace{1cm} (5.22)

Equation 5.21 becomes:

$$\frac{\partial \psi}{\partial x}(x).f(x) = \sum_{i=1}^{n} 2w_i(x - c_i) \exp\left(-\frac{(x - c_i)^2}{r_i^2}\right)f(x)$$  \hspace{1cm} (5.23)

Recall that the state vector $x$ was replaced with the lower dimension vector $\zeta \in \mathbb{R}^{n-1}$. The RBFN for the capture tube will have to be trained such that:

$$\frac{\partial \psi}{\partial \zeta}(\zeta).f_\zeta(x) = \sum_{i=1}^{n} 2w_i(\zeta - c_i) \exp\left(-\frac{(\zeta - c_i)^2}{r_i^2}\right)f_\zeta(x) = -\bar{c} + f_p(x)$$  \hspace{1cm} (5.24)

The RBFN weights will be updated using a gradient descent method:

$$w_j = w_j - 4\eta(\zeta - c_j) \exp\left(-\frac{(\zeta - c_j)^2}{r_j^2}\right)f_\zeta(x)\left(-\bar{c} + f_p(x)\right)$$

$$- \sum_{i=1}^{n} 2w_i(\zeta - c_i) \exp\left(-\frac{(\zeta - c_i)^2}{r_i^2}\right)f_\zeta(x)$$  \hspace{1cm} (5.25)

A typical way for finding solutions of differential equations using RBFN interpolation involves sampling of the state space with a grid approach technique. The RBFN is approximated using data from the sampled values and the approximation error is then calculated using Taylor series. If the basis functions are chosen on the grid [92], then the weights of the RBFN can be found by solving a linear system of equations. There are some drawbacks when approximating with sampled data, for instance, approximating the derivative can be very sensitive to errors. Also, the grid approach does not take into account the nonlinearities of the dynamical system.

Using interval methods, we can address these drawbacks and provide a guaranteed approach for finding solutions for uncertain differential equations. In interval RBFN the samples in the training
dataset are replaced with boxes and all possible system trajectories in each evaluated box are used in the training process (Figure 79).

![Figure 79. Vector field evaluation: a) Interval evaluation; b) Vertex evaluation.](image)

The main differences between a classical RBFN and an interval RBFN approach as described in this thesis are listed below:

- for the interval RBFN, the input dataset is described by intervals (box dataset). The dataset contains samples \( \{[x_j], [y_j]\} \) with \( j \in \{1, 2, \cdots, N\} \);
- the training of the interval RBFN weights is performed with gradient descent methods as in classical RBFNs, but using their interval counterparts;
- an extra training step, specific for the propagation problem described previously, is used for the interval RBFN. This step consists in training the network weights using values of the target function derivatives. In this particular case, because we have an analytical expression for \( \frac{\partial \psi}{\partial x}(x) \), using interval methods one can guarantee an approximation within a predefined accuracy;
- the output of the interval RBFN is evaluated using an inclusion function for \( \psi \), i.e. \([\psi](x)\).

In any training method for neural networks, the initialisation plays a very important role for the convergence of the algorithm. We propose to use vertex propagation in order to obtain an initial guess of the stability region which will become the starting point for RBFN training. All trajectories which exit the initial set \( \mathcal{D}_0 \) are found using interval methods and enclosed with the vertex propagation (Figure 80).

Pendulum dynamical system is proposed for testing the algorithm:

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -\sin(x_1) - 0.65x_2
\end{align*}
\]

(5.26)

The initial set \( \mathcal{D}_0 \) is represented by a 2D box \([x_0] \in [-0.8, -0.6] \times [0.4, 0.6] \).
In the second step we want to bracket the boundary of the capture tube between two approximations, inner and outer. For this purpose, two RBFNs are created, $\psi_{OUT}$ and $\psi_{IN}$, and they are initialised and trained using the vertex propagation generated in the previous step. The number of radial basis functions and their width is chosen depending on the size and distribution of the initial data. In the beginning $\psi_{OUT}$ and $\psi_{IN}$ coincide, but they are going to become the inner and outer approximation of the capture tube after they are trained using the derivative in the next step.
For the last step, the two previously created RBFNs are going to be trained using the derivative. $\psi_{OUT}$ is trained such that no trajectory can intersect the approximation from inside to outside and $\psi_{IN}$ is trained such that no trajectory can intersect the approximation from outside to inside. If these two conditions are satisfied (Eq. 5.27), then the boundary of the capture tube is bracketed between the two approximations. To achieve this, $\psi_{OUT}$ and $\psi_{IN}$ are trained such that:

$$
\frac{\partial \psi_{OUT}}{\partial \zeta}(\xi). f_{\xi}(x) - f_{p}(x) = -\bar{c} \\
\frac{\partial \psi_{IN}}{\partial \zeta}(\xi). f_{\xi}(x) - f_{p}(x) = \bar{c}
$$

Both approximations are shown in Figure 82. The distance between them depends on the magnitude of the parameter $\bar{c}$ which defines the accuracy of the approximation (the derivatives are zero on the border of the exact solution). $\bar{c}$ can be arbitrary small, but smaller training errors have to be achieved and the intervals for the training dataset have to be chosen smaller.

![Figure 82. Inner and outer approximations of the capture tube: $\psi_{OUT}$ is yellow and $\psi_{IN}$ is green.](image)

Training only the derivative of the RBFN on its own is not enough for the approximation to converge to the boundary of the capture tube. Although the training error for the derivative is decreasing the approximation will lose the dependency with the initial set $\mathcal{D}_0$ because the RBFN approximation will move in the state space. To solve this problem, the training process is improved by adding a set of anchor points $X_a \in \partial \mathcal{D}_0$. An extra training step is performed using the anchor points as direct values of the approximated function.
The training error for the derivative is shown in Figure 83. The error spikes at the beginning of the training process correspond with the training of the anchor points. These error spikes disappear when the approximation converges to the boundary of the capture tube.

![Figure 83. Training error.](image)

**Table 5.2.**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RBF BOUND (in: f, D₀, ℰ, ℰ; out: ψ_OUT, ψ_IN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>P_Init = SIVIA(D₀);</td>
</tr>
<tr>
<td>2.</td>
<td>V_Propag = Propagate(P_Init, dt, n_steps);</td>
</tr>
<tr>
<td>3.</td>
<td>Initialise(ψ_OUT, V_Propag);</td>
</tr>
<tr>
<td>4.</td>
<td>Initialise(ψ_IN, V_Propag);</td>
</tr>
<tr>
<td>5.</td>
<td>do</td>
</tr>
<tr>
<td>6.</td>
<td>Train(ψ_OUT, P_Init);</td>
</tr>
<tr>
<td>7.</td>
<td>S_OUT = SIVIA(ψ_OUT);</td>
</tr>
<tr>
<td>8.</td>
<td>TrainDerivative(ψ_OUT, S_OUT);</td>
</tr>
<tr>
<td>9.</td>
<td>Train(ψ_IN, P_Init);</td>
</tr>
<tr>
<td>10.</td>
<td>S_IN = SIVIA(ψ_IN);</td>
</tr>
<tr>
<td>11.</td>
<td>TrainDerivative(ψ_IN, S_IN);</td>
</tr>
<tr>
<td>12.</td>
<td>while dist(ψ_OUT, ψ_IN) &gt; ℰ</td>
</tr>
<tr>
<td>13.</td>
<td>return;</td>
</tr>
</tbody>
</table>
Each step of the above algorithm is detailed below:

1. **Generate a paving for the initial set** $\mathcal{D}_0$ – use interval methods (based on SIVIA algorithm) to find an enclosure for the trajectories which exit the initial set $\mathcal{D}_0$;

2. **Vertex propagation** – propagate points on the initial boundary to generate an initial guess for the shape of the capture tube;

3. **Initialise the inner and outer interval RBFNs** – the RBFN for the inner and outer approximations are initialised using the vertex propagation from the previous step;

4. **Train inner and outer RBFNs** – the network weights are trained using the direct method;

5. **Get the inner and outer interval RBFN outputs** – the output of the RBFNs is evaluated using interval methods (SIVIA algorithm);

6. **Train inner and outer RBFNs using derivatives** – the network weights are trained using the derivative formula for weight adjusting (5.22). For this step the training dataset contains intervals;

7. **Exit condition** – the algorithm exits when the distance between the inner and outer approximations is smaller than a predefined epsilon;

### 5.4 ODE solution boundary approximation using linear interpolation

#### 5.4.1 General aspects

In the Section 5.3 solutions for uncertain ODEs were approximated using RBFNs. Although they are easy to generalise to multiple dimensions, they can interpolate complex functions and their computation does not require complex algorithms, RBFNs also have some drawbacks. One issue for the gradient descent methods which is used for training RBFN is that they can converge to a local minimum. Another issue arises from the fact that in the previous section RBFNs were combined with interval methods to achieve a guaranteed estimation of the boundary of the capture tube. Interval methods can give guaranteed solutions to mathematical problems, but at the same time, they can be overestimated. When dealing with functions that have complex expressions, multiple occurrences of the function’s variables can lead to overestimation. For this reason, when using interval methods, it is desirable to try and reduce the number of multiple variable occurrences as much as possible. The problem with RBFNs is that their accuracy relies on a big number of basis functions and multiple occurrences of the variables cannot be avoided. To decrease the overestimation, the RBFNs will have to be trained and evaluated using small intervals which can become computationally expensive.
There are many methods for interpolating nonlinear functions, but finding their parameters in order to solve ODEs is not always straightforward. Moreover, when combined with interval methods they can lead to the same problems as the RBFNs. From this perspective, to simplify the approach, linear interpolation could be an alternative to avoid these problems.

There are multiple reasons why linear interpolation can be better suited to be combined with interval methods:

- because the equation for the interpolation is simple, we avoid the overestimation problem when it is combined with interval methods;
- we use vertex propagation as an initial guess for the solution. In this context the interpolating function can be immediately computed;
- linear interpolation has some drawbacks (large approximation errors), but they can be addressed by using interval methods.

ODE solutions have been studied from a topological perspective as early as 1954 [97] [98], but these approaches were focused more on the analytical side. For practical purposes, numerical approaches can give solutions to a more general set of problems, but they have the drawback of not being guaranteed. As a numerical solution, a moving mesh approach has been studied and employed to solve partial differential equations [99] [100] [101]. This approach requires the usage of a vertex adjusting function which has to converge to the solution of the differential equation. Finding such a function is not trivial and its convergence can be problematic in areas with high variations in the solution. The mesh also needs to be tested using a monitor function to control vertex distribution and solution accuracy. These problems can be solved in a more general manner by using interval techniques such as interval evaluation and bisection.

5.4.2 Algorithm for generating the boundary of the ODE solution using linear interpolation

Because the boundary of the capture tube can have any shape, the linear interpolation will have to use triangulation to approximate this shape. Delaunay triangulation [102] is most commonly used to generate a convex hull in multiple dimensions. Fast algorithms have already been implemented (Quickhull [103]) and can be used to generate a convex hull for an arbitrary number of points, regardless of their distribution in space.

In a similar manner with the RBFN approach previously described, the algorithm will have the following steps:
1. **Generate a paving for the initial set** $\mathcal{D}_0$ - use interval methods to find an enclosure for the trajectories which exit the initial set $\mathcal{D}_0$;

2. **Euler vertex propagation** – propagate points on the initial boundary to generate an initial guess for the shape of the capture tube;

3. **Capture tube boundary mesh** - use a convex hull algorithm to generate a triangulation of the vertex propagation, then find the boundary of the convex hull. This hypersurface will become the approximation of the capture tube boundary;

4. **Compute a linear interpolation function** - for each simplex in the boundary, compute the parameters of the linear interpolation function;

5. **Check the approximation error on each simplex** - If the error is bigger than a predefined threshold, bisect the simplex. If the error is smaller than the threshold, join two adjacent simplexes;

6. **Adjust all the simplexes on the hypersurface** – all simplexes are adjusted such that the cross-out condition is not satisfied.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>LINEARBOUND (In: $f, \mathcal{D}<em>0, \bar{c}, \varepsilon$; Out: $\psi</em>{\text{OUT}}, \psi_{\text{IN}}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\mathcal{B}_{OB} =$SIVIA$(\mathcal{D}_0)$;</td>
<td></td>
</tr>
<tr>
<td>2. $\mathcal{B}<em>{\text{out}}^0 = \mathcal{B}</em>{OB};$ $\mathcal{B}<em>{\text{in}}^0 = \mathcal{B}</em>{OB};$</td>
<td></td>
</tr>
<tr>
<td>3. do</td>
<td></td>
</tr>
<tr>
<td>4. $\mathcal{B}<em>{\text{out}}^{k+1} = \text{Propagate}(\mathcal{B}</em>{\text{out}}^k, dt);$ $\mathcal{B}<em>{\text{in}}^{k+1} = \text{Propagate}(\mathcal{B}</em>{\text{in}}^k, dt);$</td>
<td></td>
</tr>
<tr>
<td>5. $\mathcal{S}<em>{\text{out}}^k = \text{Mesh}(\mathcal{B}</em>{\text{out}}^k, \mathcal{B}<em>{\text{out}}^{k+1});$ $\mathcal{S}</em>{\text{in}}^k = \text{Mesh}(\mathcal{B}<em>{\text{in}}^k, \mathcal{B}</em>{\text{in}}^{k+1});$</td>
<td></td>
</tr>
<tr>
<td>6. $\mathcal{G}<em>{\text{out}}^k = \text{Interpolate}(\mathcal{S}</em>{\text{out}}^k);$ $\mathcal{G}<em>{\text{in}}^k = \text{Interpolate}(\mathcal{S}</em>{\text{in}}^k);$</td>
<td></td>
</tr>
<tr>
<td>7. $\delta_{\text{out}}^i = \text{Error}(\mathcal{G}<em>{\text{out}}^k, \mathcal{S}</em>{\text{out}}^i);$ $\delta_{\text{in}}^i = \text{Error}(\mathcal{G}<em>{\text{in}}^k, \mathcal{S}</em>{\text{in}}^i);$</td>
<td></td>
</tr>
<tr>
<td>8. If $\delta_{\text{out}}^i &gt; \varepsilon_B$ then Bisect($\mathcal{S}_{\text{out}}^i$);</td>
<td></td>
</tr>
<tr>
<td>9. If $\delta_{\text{out}}^i &lt; \varepsilon_I$ then Join($\mathcal{S}_{\text{out}}^i$);</td>
<td></td>
</tr>
<tr>
<td>10. If $\delta_{\text{in}}^i &gt; \varepsilon_B$ then Bisect($\mathcal{S}_{\text{in}}^i$);</td>
<td></td>
</tr>
<tr>
<td>11. If $\delta_{\text{in}}^i &lt; \varepsilon_I$ then Join($\mathcal{S}_{\text{in}}^i$);</td>
<td></td>
</tr>
<tr>
<td>12. Adjust($\mathcal{G}<em>{\text{out}}^k$); Adjust($\mathcal{G}</em>{\text{in}}^k$);</td>
<td></td>
</tr>
<tr>
<td>13. while $\text{dist}(\mathcal{G}<em>{\text{out}}^i, \mathcal{G}</em>{\text{in}}^i) &gt; \varepsilon$</td>
<td></td>
</tr>
<tr>
<td>14. return;</td>
<td></td>
</tr>
</tbody>
</table>
All practical and theoretical aspects of the steps presented above are detailed in the rest of this section. The illustrative examples are provided for a third order system and then the method is generalised for multiple dimensions.

Let us consider a third order dynamical system:

\[
\begin{align*}
\dot{x}_1 &= -x_2 x_3 + 1 \\
\dot{x}_2 &= x_1 x_3 - x_2 \\
\dot{x}_3 &= x_3^2 (1 - x_3)
\end{align*}
\]  
(5.28)

5.4.2.1 Generate a paving for the initial set \( \mathcal{D}_0 \)

A paving for the outbound areas on the initial set \( \mathcal{D}_0 \) can be generated using a SIVIA algorithm. The enclosure of this paving will be used to provide the initial vertices, needed for the Euler vertex propagation in the next step. In Figure 84, the initial set is represented by a 3D box. The initial set can have any arbitrary shape as long as SIVIA algorithm is used to generate a paving.

For an autonomous dynamical system \( \dot{x} = f(x) \) the outbound set on the boundary of \( \mathcal{D}_0 \) is defined as:

\[
B_{OB} : \{ x \mid x \in \partial \mathcal{D}_0, \frac{\partial b(x)}{\partial x} f(x) \geq 0 \}
\]  
(5.29)

where \( b(x) \) is a given function which characterises the initial set \( \mathcal{D}_0 \).

Figure 84. Paving of the initial set and outbound trajectories.
5.4.2.2 Euler vertex propagation

The outbound set $\mathcal{B}_{OB}$ needs to be propagated in order to generate an initial guess for the capture tube approximation. For this purpose, a set of points is chosen on the boundary of $\mathcal{B}_{OB}$:

$$x_i^B \in \partial \mathcal{B}_{OB}, \ i \in \{1, \ldots, n_B\}$$  (5.30)

where $n_B$ is the number of boundary points to be propagated.

Euler method is used next to generate the vertex propagation:

$$x_{i,k+1}^B = x_{i,k}^B + \Delta t \cdot f(x_{i,k}^B), \ i \in \{1, \ldots, n_B\}$$  (5.31)

A more precise integration method is not needed as long as we are going to use linear interpolation. Integration step size $\Delta t$ is adjusted using interval methods such that the integration does not become unstable. In order to compute the approximation error of the vertex propagation, we define the interval hull of two points:

$$[x_{i,k}^B] = [x_{i,k}^B, x_{i,k+1}^B]$$  (5.32)

With this interval we can evaluate the approximation error as follows:

$$\delta_k = \max_i (1 - |f(x_{i,k}^B)| \cdot |f([x_{i,k}^B])|)$$  (5.33)
5.4.2.3 Capture tube boundary mesh

An appropriate mesh has to be found to characterise the boundary of the capture tube. We want to have a guaranteed bracketing of the capture tube boundary and this mesh will be adjusted to find the inner and outer approximations. To satisfy these constraints, the mesh needs to have the following properties:

- the mesh will enclose all points propagated in the previous step;
- the mesh cannot have any gaps as this will invalidate the guaranteed approach;
- the mesh will have a shape as close as possible to the system trajectories because this will have a high impact on the adjustment process and computation time;
- mesh generation should not be constrained by the number of dimensions;

A general approach that works in multiple dimensions and can satisfy the above requests is Delaunay triangulation applied to the vertex propagation in order to find the convex hull. The Quickhull algorithm [103] is used for this, as it can work for a high number of dimensions. Figure 87 shows the convex hull of all propagated vertices obtained with the Quickhull algorithm. The obvious problem of this approach is that the convex hull does not give an approximation which is close to the initial guess. This issue comes from the fact that the propagation does not have a convex shape. There are some algorithms to compute a hull for non-convex shapes, but some problems can arise from the fact that the solution for a concave hull is not unique. Some approaches for this algorithms use alpha shapes [104] or k-nearest neighbours [105], but they all depend highly on the data density and they

---

*Figure 86. Euler Vertex propagation for the initial set.*
are sensitive to variations in the data structure. This leads again to problems because the generated concave hull is not close to the system trajectories in most of the cases.

![Convex hull of the vertex propagation.](image1)

**Figure 87. Convex hull of the vertex propagation.**

The best solution is to use a step-by-step approach, which is a combination between the convex hull and vertex propagation. The convex hull is generated incrementally for each step of the propagation, as it is characteristic for every Lagrangian approach. In this way, the hull can be as close as possible to the system trajectories and the adjusting process will also be shorter.

![Non-Convex hull of the vertex propagation.](image2)

**Figure 88. Non-Convex hull of the vertex propagation.**
The generated mesh is a hypersurface in $\mathbb{R}^n$ defined by $n_s$ simplexes:

$$\mathcal{S} = \{S_1, S_2, \ldots, S_{n_s}\} \quad (5.34)$$

### 5.4.2.4 Compute a linear interpolation function

The linear interpolation function is, in fact, the equation of a hyperplane in $\mathbb{R}^n$ and it has the following form:

$$g: \mathbb{R}^n \to \mathbb{R}, \quad g(x) = N_h x + d \quad (5.35)$$

where:

- $g(x)$ is the linear interpolation function for approximating the boundary of the capture tube;
- $N_h$ is the normal vector of the hyperplane $(n_1^h, n_2^h, \ldots, n_n^h)$;
- $d$ is the distance from hyperplane to origin of the system of coordinates;

Each simplex $S_j$ in the mesh computed at the previous step is defined by $n$ points:

$$p_i \in \mathbb{R}^n, \quad i \in \{1, \ldots, n\}$$

With these $n$ points we are able to compute $n - 1$ vectors:

$$v_j \in \mathbb{R}^n, \quad j \in \{1, \ldots, n - 1\} \quad (5.36)$$

$$v_j = \frac{p_{j+1} - p_j}{\|p_{j+1} - p_j\|}$$

The cross product in $\mathbb{R}^n$ of this set of vectors is:

$$v_1 \times v_2 \times \cdots \times v_{n-1} = \begin{vmatrix} \hat{e}_1 & \cdots & v_1^{n-1} & v_2^{n-1} \\
\vdots & \ddots & \vdots & \vdots \\
v_{n-1}^{n-1} & \cdots & \hat{e}_n & \cdots & v_n^{n-1} \end{vmatrix} \quad (5.37)$$

where $(\hat{e}_1, \ldots, \hat{e}_n)$ are the basis vectors in $\mathbb{R}^n$.

The cross product is perpendicular on all its arguments which makes it perpendicular on the hyperplane as all the $n - 1$ vectors are also contained in the hyperplane:

$$N_h = v_1 \times v_2 \times \cdots \times v_{n-1} \quad (5.38)$$

Next, the distance from hyperplane to the origin is:

$$d = -N_h \cdot p_i, \quad \forall i \in \{1, \ldots, n\} \quad (5.39)$$
Now we are able to define a linear interpolation function for each simplex $s_j$ in the mesh:

$$\mathbb{G} : \{ \varphi_j : S_j \to \mathbb{R} \mid \varphi_j(x) = N_h x + d_j \}$$

(5.40)

5.4.2.5 The approximation error introduced by each simplex

To control the accuracy of the capture tube approximation, we need to be able to change the shape and resolution of the generated mesh. The following two steps are needed:

- estimate the approximation error for each simplex on the mesh;
- reduce or increase the number of simplexes if the error is too small or too big;

For error estimation a paving is generated for the simplex using the SIVIA algorithm for the following constraint:

$$N_h x + d = 0$$

(5.41)

The algorithm returns a set of boxes $\{x_i^s\} \in \mathbb{R}^n, i \in \{1, \ldots, n_B\}$ where $n_B$ is the total number of boxes. The following relation is used to evaluate the error on the simplex:

$$\delta_s = \text{width} \left( \bigcup_{i=1}^{n_B} N_h \cdot [f](x_i^s) \right)$$

(5.42)

![SIVIA generated paving of a simplex (the simplex is triangle in 3D).](image)

Figure 89. SIVIA generated paving of a simplex (the simplex is triangle in 3D).
If $\delta_s$ is bigger than a predefined threshold, then the accuracy of the estimation must be increased. To solve this, mesh resolution will have to be increased through bisections. A simplex is bisected by adding a point on one of its edges.

If $\delta_s$ is smaller than a predefined threshold, then the accuracy of the estimation has to be decreased, because the mesh complexity is too high. To solve this, mesh resolution will have to be decreased by joining two adjacent simplexes. One vertex is not propagated and the algorithm joins the simplexes (for instance, four triangles will become three in the next step).

![Figure 90. Mesh accuracy control: a) Simplex bisection; b) Simplex union.](image)

5.4.2.6 Adjust all the simplexes on the hypersurface

To be able to guarantee the validity of the mesh approximation for the capture tube, we need to adjust each simplex $S_j$ on the mesh such that its associated linear function $g_j$ does not satisfy the cross-out condition:

$$\begin{cases} g_j(x) = 0 \\ \frac{\partial g_j}{\partial x}(x) > 0 \end{cases} \quad (5.43)$$

From the RBFN approach recall Equation (5.18), we have to compute the derivative such that:

$$\frac{\partial g_j}{\partial x}(x) = -c, \quad c > 0 \quad (5.44)$$

For each simplex $S_j$ we use SIVIA to compute a paving for the following constraint:

$$g_j(x) = 0$$

The paving contains the boxes $[x_{i,j}^x] \in \mathbb{R}^n, i \in \{1, \ldots, n_B\}$. We are able to write the derivative equation in interval form:
From Equation (5.35) we have:

\[ g_j(x) = \mathcal{N}_h \cdot x + d_j \Rightarrow \frac{\partial g_j}{\partial \mathbf{x}}(\mathbf{x}) = \mathcal{N}_h' \cdot \dot{x} \]  

Equation (5.45) becomes:

\[
\begin{bmatrix} F'_s \end{bmatrix} = \bigcup_{i=1}^{n_B} [f](\mathbf{x}^i_j) \\
\mathcal{N}_h' \cdot \begin{bmatrix} F'_s \end{bmatrix} = [-c, 0]
\]

We need to estimate a new set of parameters, \( \mathcal{N}_{\hat{h}} = (\hat{n}_1, \hat{n}_2, \ldots, \hat{n}_n) \), for the hyperplane such that equation 5.47 is satisfied. Because we have \( n \) unknown parameters, we have to add another set of \( n - 1 \) constraints such that the system of equations becomes fully determined. For this, we impose the constraint that only one point in the simplex will change with each adjusting step.

Recall that each simplex \( S_j \) has \( n \) points:

\[
S_j: \{p_1, p_2, \ldots, p_n\} \quad \text{(5.48)}
\]

The point to be adjusted, \( p_a \), is selected from the simplex based on the integration time associated with each point. The propagation time for each simplex is:

\[
T_s: \{t_1, t_2, \ldots, t_n\} \\
\hat{a} = \arg \max_{1 \leq i \leq n} (t_i) \quad \text{(5.49)}
\]

With the remaining \( n - 1 \) points \( p_i, i \in \{1, 2, \ldots, a - 1, a + 1, \ldots, n\} \) we can construct \( n - 2 \) vectors:

\[
v_j \in \mathbb{R}^n, \quad j \in \{1, \ldots, n - 2\} \\
v_j = \frac{p_{j+1} - p_j}{\|p_{j+1} - p_j\|}, \quad j \neq a \quad \text{(5.50)}
\]

With these vectors, we can construct an initial interval approximation for \( \mathcal{N}_{\hat{h}}' \):

\[
\begin{bmatrix} \mathcal{N}_{\hat{h}}' \end{bmatrix} = v_1 \times v_2 \times \cdots \times v_{n-2} \times \begin{bmatrix} F'_s \end{bmatrix} \quad \text{(5.51)}
\]
To contract this approximation, we can use the following CSP:

\[ \mathcal{H}_n; \ h(x) = \begin{cases} x \cdot [F^l_j] > -c \\
 x \cdot [F^l_j] < 0 \end{cases}, \quad h: \mathbb{R}^n \to \mathbb{R}^2 \] (5.52)

The solution set for the above CSP is:

\[ \mathbb{H}_{OUT} = \{ x \in [\mathcal{N}_h^l] \mid h(x) = [-c, 0] \} \] (5.53)

In order to have a normal vector \( \mathcal{N}_h^l \) for the simplex \( S_j \) that is guaranteed to satisfy Equation (5.43) we have to find an inner approximation for the set \( \mathbb{H}_{OUT} \). To achieve this, we define the a complementary CSP for \( \mathcal{H}_n \):

\[ \mathcal{H}_n^\bar{c}; \ h(x) = \begin{cases} x \cdot [F^l_j] < -c \\
 x \cdot [F^l_j] > 0 \end{cases}, \quad h: \mathbb{R}^n \to \mathbb{R}^2 \] (5.54)

A forward-backward contractor can be used to contract the domain of \( x \):

\[ \mathcal{C}_{\cdash} (h, [\mathcal{N}_h^l], (-\infty, -c] \cup [0, \infty)) \Rightarrow x \in [\mathcal{N}_h^l] \] (5.55)

The inner approximation of the set \( \mathbb{H} \) is then found:

\[ [\mathcal{N}_h^l] = [\mathcal{N}_h^l] \setminus [\bar{N}_h^l] \] (5.56)

Now we have:

\[ \begin{cases} \frac{\partial g_j}{\partial x}(x) > -c \\
 \frac{\partial g_j}{\partial x}(x) < 0 \end{cases}, \quad \forall \bar{N}_h^l \in [N_h^l] \] (5.57)

With the above CSP solved, we now have a new normal vector to the simplex. The remaining parameter \( \hat{d} \) is straightforward to compute:

\[ \hat{d} = -\bar{N}_h^l \cdot p_i, \quad \forall i \in \{1, \ldots, n-1\} \setminus \{a\} \] (5.58)

The point to be adjusted is computed as follows:

\[ \hat{x}_a = x_a - (\bar{N}_h^l \cdot x_a + \hat{d}) \] (5.59)
To bracket the capture tube, we need to compute an outer and an inner approximation. The outer approximation is adjusted with the CSP described above in (5.52) and the inner is adjusted in similar manner with the following CSP:

$$H_n : h(x) = \begin{cases} x \cdot [F_S]^l < c \\ x \cdot [F_S]^l > 0 \end{cases}, \quad h: \mathbb{R}^n \rightarrow \mathbb{R}^2, c > 0$$

(5.60)

And the solution set for adjusting the inner approximation is:

$$\mathbb{H}_{IN} = \{ x \in \tilde{N}_h \mid h(x) = [-c, 0] \}$$

(5.61)

Finally, the two hypersurface approximations will become inner and outer brackets for the capture tube:

$$S_{out} = \{ S_{out}^1, S_{out}^2, \ldots, S_{out}^{n_{out}} \}$$

$$S_{in} = \{ S_{in}^1, S_{in}^2, \ldots, S_{in}^{n_{in}} \}$$

(5.62)

5.4.2.7 Case study

The algorithm is tested with two dynamical systems, a second order system and a third order system.

The second order system is the pendulum:

$$\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -\sin(x_1) - 0.15x_2
\end{align*}$$

(5.63)

System trajectories are bracketed for a time interval \( t \in [0, 6] \) and the initial box \( \mathbf{x} \in [0.6, 0.8] \times [0.4, 0.6] \).
Next, we test the algorithm on a third order system:

\[
\begin{align*}
\dot{x}_1 &= -x_2 x_3 + 1 \\
\dot{x}_2 &= x_1 x_3 - x_2 \\
\dot{x}_3 &= x_3^2 (1 - x_3)
\end{align*}
\] (5.64)

In this case, adjusting the mesh is more complex because going to higher dimensions increases not only the required computations but also adds extra degrees of freedom to the simplexes. A simplex in \( \mathbb{R}^n \) will have \( n - 2 \) degrees of freedom which means in two dimensions the solution for the adjusting process is unique, but with three dimensions, there is one extra degree of freedom for the simplex to be adjusted.

The system is propagated for a time interval \( t \in [0,1] \) and the initial box \( x \in [-6,-5] \times [5,6] \times [2,3] \). The distance between the inner and outer approximations can be controlled by adjusting the mesh accuracy.
Another test is performed with the initial box in a different zone of the state space. System trajectories have more variations in this region, which makes the approximation problem more difficult.

The system is now propagated for a time interval $t \in [0,4]$ and the initial box $\mathbf{x} \in [-6,-3] \times [3,6] \times [0.6,2]$. The propagation algorithm is able to give a tight enclosure for system trajectories.
A case of particular interest is when the initial box contains the equilibrium point. Now there are multiple exit regions on the initial box and all the system trajectories will return inside the initial box after a finite amount of time.

The initial box is chosen such that the equilibrium point is inside $\mathbf{x} \in [0, 2] \times [0, 2] \times [1.5, 2]$. In this case, the propagation finishes when all the trajectories return inside the initial box.

Figure 95. Propagation with an equilibrium point inside the initial box.
Figure 96. Propagation finishes when all trajectories return to the initial box. There are multiple exit points on different facets of the initial box.

5.4.2.8 Comparison with the RBF network approximation

The linear interpolation algorithm was compared with the RBF network approximation for a second-order system. A comparison for a third-order dynamical system could not be conducted because the RBFN boundary approximation algorithm computation time increased exponentially.

For a second-order system, both algorithms were tested for 1.5s propagation time and the same precision for the inner and outer brackets. The computation time for the RBFN approach was 105s and for the linear interpolation approach 1.1s. The tests were conducted on a laptop with an Intel i7-8750h processing unit.

The RBFN propagation is a lot slower because of the training process which is computationally expensive. The propagation results are shown in the two figures below.
5.4.2.9 Comparison with guaranteed integration solver Dynibex

To validate our approach, we compare it with the latest solver for guaranteed integration using interval methods, Dynibex [15] [106], which can propagate high dimension systems with uncertainty in the parameters and initial domain.

The first comparison illustrates the performance of both algorithms when the second order pendulum system has uncertainty in the initial domain, but no parametric uncertainty. The grey box is the result produced by the Dynibex algorithm for 7 seconds of propagation of the initial box.

In this particular situation, Dynibex is able to give a tight enclosure, comparable with our algorithm for the given propagation time. Next, the propagation is performed for the pendulum system with additive and parametric uncertainties in the system equations.
The pendulum with parametric and additive uncertainty:

\[
\begin{align*}
\dot{x}_1 &= x_2 + \gamma \\
\dot{x}_2 &= -\sin(x_1) - F_R x_2 + \gamma
\end{align*}
\]  

(5.65)

where \( \gamma \in [-0.001, 0.001] \) and \( F_R \in [0.15, 0.2] \)

Figure 99 shows the results produced by both algorithms when the system with uncertainty is simulated for 7 seconds. The algorithm proposed in Section 5.4 can give a tight enclosure for the trajectories and also give a bracketing for the manifold boundary. The Dynibex algorithm has produced an unsatisfactory result in this case (grey box), which shows that the approach is not suitable to handle ODE with parametric uncertainties.

![Figure 99. Comparison with Dynibex for the system with parametric uncertainty.](image)

Next comparison is performed for the third order dynamical system with different initial domains. Figure 100 shows the comparison between the two algorithms for a 2 second propagation with the initial box \( x \in [-6, -5] \times [5, 6] \times [2, 3] \). In this case, Dynibex algorithm is able to give tight enclosure (red box), comparable with the result produced by our algorithm.
Another comparison is performed when the initial box is in a different region of the state space. In this case, the propagation is done for 0.4 seconds for the initial box \( \in [-6, -3] \times [3, 6] \times [0.6, 2] \). Because of the big nonlinearities in that region of state space, the Dynibex algorithm gives a highly overestimated result even for small propagation times. This can be addressed by using bisections, but with a high computational cost.

Figure 100. Comparison with Dynibex for a third order system

Figure 101. Comparison with Dynibex when the initial box is located in a more complex region of the dynamical system
5.5 Conclusion

This chapter proposed new methods for estimating and bracketing the solution manifold for ordinary differential equations with uncertainties. The proposed methods are part of the Lagrangian and Eulerian approaches described in Chapter 2, Section 2.4. Multiple ways to find ODE solutions have been proposed in the past (see Chapter 2, Section 2.7), but they have some drawbacks which have been addressed in this chapter, such as:

- The integration algorithm becomes unstable after several steps (overestimation increases unboundedly);
- Some algorithms are able to provide a stable inner approximation, but the outer guaranteed approximation is not provided;
- There is no method for finding the approximation error (inner and outer brackets);
- Some algorithms require bisections to be able to deal with uncertainty which makes them computationally expensive.

The methods described in this chapter rely on using interval methods to find estimations for the solution boundary of ODEs with uncertain initial condition. In our approach, the boundary is characterised by multidimensional hypersurfaces. To be able to check that the hypersurface approximates the capture tube in a guaranteed way, an interval methods test, called cross-out condition, is used. This test requires the hypersurface to be defined as a parametrised multidimensional function. Two methods to generate this function are proposed in this chapter: the first method uses radial basis function networks (Eulerian method); the second method uses vertex propagation and linear interpolation (Lagrangian method).

The RBFNs were the first option to be investigated for approximating the boundary of ODE solutions because they have some clear advantages:

- can be trained to achieve an arbitrary small approximation error;
- can be easily extended to multiple dimensions;
- combined with interval methods they can provide guaranteed brackets;

The RBFN used in this chapter combines the classical training algorithm with an additional training step which uses interval methods to estimate the output of the RBFN and trains the derivative of the network. This extra step is required because in our case when finding ODE solutions, we have an analytical expression for the derivative. By using interval methods, we can guarantee that if the training process converges to the desired approximation error then the approximation satisfies the cross-out condition.
An RBFN was trained to approximate a solution for a second order nonlinear dynamical system, but some drawbacks were encountered in the process. The training process is computationally expensive and this becomes a burden when using interval methods. The RBFN function relies on multiple basis functions for an accurate approximation which becomes a problem when dealing with interval methods. Multi incidence of the variables in the function expression leads to overestimation when using interval methods.

The last part of the chapter proposes a new approach to overcome the computational cost issue encountered in the previous method. Vertex propagation can be used to generate an inner approximation for ODE solutions. Because this method does not have overestimation, we proposed a combination of this approach with interval methods to generate guaranteed approximations. A triangulation step is used to generate an approximating mesh which serves as a linear interpolation in the next step. A linear function for the triangulation is used to test the cross-out condition and adjust the mesh to match the ODE solution.
Chapter 6: Conclusions and future work

6.1 Closure

This thesis focused on finding new methods to study the stability of nonlinear dynamical systems in the presence of uncertainties. This topic is of particular interest because of multiple reasons:

- While there exist analytical solutions to prove the stability for linear systems, this is not, in general, the case for nonlinear systems. Moreover, if the linear system is with uncertainty, the stability problem becomes more difficult;
- The most used stability theory for nonlinear systems (Lyapunov stability) can be applied only for particular cases, i.e. for mechanical systems;
- More general stability theory exists (positive invariance proposed by Nagumo), but there is a lack of analytical or numerical methods to find the tangent cone;
- A mismatch between mathematical models and the real systems is unavoidable. There is a need to develop stability theory to handle uncertainties in the models.

All problems listed above were addressed throughout the thesis, mostly by using interval methods combined with the existing theory.

A differential drive robot is studied first, and a sliding mode control law is developed for the system. The control law is derived using multiple models of the robot and FTC is also tested. Lyapunov stability was used to develop new fault-tolerant control techniques, as well as novel control strategies. To the best of my knowledge, in this thesis, current sensors to control the torque of actuators (dc motors) for a mobile robot were proposed for the first time.

A new active FTC technique was proposed for the SMC based on kinematic model, which was enhanced by adding a fault hiding block in the control structure. Fault hiding method is an active FTC approach which allows for fault compensation without retuning the controller.

The SMC is robust against disturbances and uncertainties in the model, but it has some drawbacks: there is no method for tuning the controller parameters; it is not possible to estimate the degree of robustness and controller tolerance to disturbances. Interval methods were employed to successfully address these issues. Methods for tuning the SMC parameters and for estimating the controller robustness were proposed for the first time.

The stability of nonlinear dynamical systems was analysed from the Eulerian perspective in the next part of the thesis. Because there is a lack of methods in the literature for stability analysis of
uncertain dynamical systems from this perspective, interval analysis techniques were used to address it. Lyapunov based approaches are used most of the times to prove the stability of nonlinear systems, but they only work in particular cases and cannot deal with uncertainties. Based on LaSalle’s invariance principle, a new automatic method for finding positive invariant sets for nonlinear dynamical systems is proposed. It was shown that by adding small perturbations to the energy function of nonlinear mechanical systems we can eliminate the problematic points where the Lyapunov function derivative is negative semi-definite or positive. This method is shown to be able to prove the stability of various nonlinear dynamical systems with uncertainties. Interval methods were also used to estimate parameters for the perturbation functions in a guaranteed way. In some cases, they can also be used to find Lyapunov function candidates when an energy function of the system is not available.

Finding solutions for ODEs with uncertain initial condition was addressed in the Chapter 5. Multiple methods to solve this type of problems were proposed in the past, but they all have some drawbacks, such as: the integration algorithm becomes unstable after several steps, there is no method for finding the approximation error, high computational cost.

The newly proposed methods rely on using interval methods to find estimations for the solution boundary of ODEs with uncertain initial condition, which is described by hypersurfaces. To be able to check that the hypersurface approximates the capture tube in a guaranteed way, an interval methods test, called cross-out condition, is used.

Two approaches were proposed to approximate the ODE solution boundary:

- RBFN approximation;
- Vertex propagation and linear interpolation.

RBFNs were investigated first because they have some advantages when approximating multidimensional functions. They are part of artificial neural networks and have training algorithms that can provide arbitrary small approximation errors. The novelty proposed in this thesis was to include an extra step that relies on interval methods in the training process. However, this new training method requires high computational time. This leads to further investigation to find faster algorithms which were described in the last part of the thesis. Vertex propagation was used to generate an initial guess for the ODE solution boundary. A triangulation method was then applied to generate an approximating mesh which is used for linear interpolation. By deriving a linear function for the triangulation we can use interval methods in conjunction with the cross-out condition and adjust the mesh to match the ODE solution. A bracketing method was proposed to provide inner and outer
approximations of the solution manifold. The algorithms were tested on second and third order dynamical systems commonly used in the literature.

6.2 Future work

Future work plans include the extension of the work conducted in Chapter 4 for autonomous nonlinear systems to time-variant systems. Stability analysis for this class of systems is difficult because it is not easy to find Lyapunov functions for them. Barbalat’s lemma [5] has been used to prove the stability for such systems, but it needs some assumptions about the bounds of the time-variant system. More complex systems are being studied using interval methods, such as: robotic sailboat system with time-varying trajectory and wind uncertainty, robotic sailboat system with time-varying trajectory and uncertainty in the velocity [20].

Artificial neural networks are currently a hot topic in the computer science research community. They have been shown to give remarkable results in topics like image processing, voice recognition, vehicle autonomy, robotics, etc. The work conducted in this thesis which involves RBFNs combined with interval methods can be further extended to more complex artificial neural networks. The high computational cost can be addressed by employing parallel computing techniques which are regularly used for training artificial neural networks.

Even though the methods for finding ODE solutions provided in Chapter 5 were developed for multiple dimensions, the solver used for the test cases is only able to deal with two and three dimensions. Future work includes the extension of the current algorithms to multiple dimensions and testing if they are computationally feasible.
List of Publications

Journal Articles


Conference Proceedings


References


