

Brand Protection for Steel Packaging Products

ANNA TOUTOUNTZI

*Submitted to Swansea University in fulfilment of the requirements
for the Degree of Doctor of Engineering*



Swansea University
Prifysgol Abertawe

Swansea University
2024

Dedicated to my yaya Anna who taught me that "the wolf's neck is thick because he does things on his own", my pappou Niko for warning me not to become like my mother, my mother Elpi who made me worse than herself but gave me wings, and my uncle Achillea for being the wind beneath my wings every chance he got.

– I love you to death, my little ones.

But I'm late for a party themed "Life".

Abstract

In recent years, the underground market of counterfeit products has grown into a global network, causing the raised concern of the general public and initiating a series of reforms in governmental regulations and policies worldwide. As the largest independent metal decorating business in the UK, Tinmasters is at the centre of these developments. The overall aim of this project was the development of a novel anti-counterfeiting technology that is compatible with Tinmasters' manufacturing process, food contact/safe, and preferably overt, with a special focus on aesthetic appeal. A review of pre-existing technologies revealed a trend toward systems relying on the fast-growing capacity of wireless internet and smartphone devices. The latest anti-counterfeiting systems are track-and-trace enabled and offer user-based product authentication. The review narrowed the scope of the project to the development of a scheme for the creation of printable 2D codes, capable to store information that can be retrieved using a smartphone device. The core element of the feature is a trajectory of a 3D nonlinear dynamical system operating within its chaotic region, which is captured by the system's "*strange*" *attractor*. These types of trajectories are known for their high complexity and thought, by many, to possess beauty. More importantly, they can be retrieved via a mechanism known as *chaotic synchronisation*. In order to create a printable code, a 3D chaotic trajectory is projected to two dimensions. The printed feature is captured by a smartphone camera and is subsequently processed in order to retrieve the trajectory. An almost equally important element of the feature is a frame, especially designed to address matters of alignment, perspective correction, and coordinate transformations. Aside from the main field of nonlinear dynamics, the proposed scheme makes use of concepts and methods from the fields of image processing, digital photography, and numerical analysis.

Declarations and Statements

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

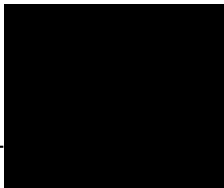
Signed: ANNA TOUTOUNTZI



Date: 08/04/2024

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

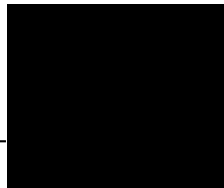
Signed: ANNA TOUTOUNTZI



Date: 08/04/2024

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

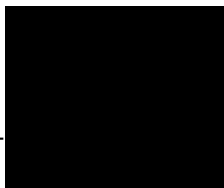
Signed: ANNA TOUTOUNTZI



Date: 08/04/2024

The University's ethical procedures have been followed and, where appropriate, that ethical approval has been granted.

Signed: ANNA TOUTOUNTZI



Date: 08/04/2024

Contents

| | |
|---|-----------|
| Abstract | i |
| Declarations and Statements | iii |
| Contents | v |
| Acknowledgements | ix |
| List of Figures | xi |
| List of Tables | xiii |
| Acronyms & Abbreviations | xv |
| 1 Introduction | 1 |
| 1.1 Thesis Objectives & Main Focus | 2 |
| 1.2 The Counterfeiting Problem | 4 |
| 1.3 Overview & Evaluation of Anti-Counterfeiting Technologies | 7 |
| 1.3.1 Project Objectives from an Industrial Perspective | 9 |
| 1.3.2 Candidate Technologies & a Scheme for Their Evaluation | 10 |
| 1.3.3 A Quick Look at Recent Trends | 14 |
| 1.4 Proposed Scheme: The NLCode | 18 |
| 1.4.1 Structure & Name Origin | 19 |
| 1.4.2 Operation Principle | 21 |
| 2 Theoretical Background: Nonlinear Dynamics & Chaos | 23 |
| 2.1 Solution of a System of ODEs | 24 |
| 2.2 Volume Deformation in State Space | 28 |
| 2.3 Equilibrium Solutions & Local Stability | 32 |
| 2.4 Lyapunov Spectrum | 36 |
| 2.5 Attracting Sets, Attractors & Basins of Attraction | 41 |
| 2.6 Chaos & Strange or Chaotic Attractors | 42 |
| 2.7 Chaotic Synchronisation & Homogeneous Driving | 47 |
| Part I Methodology: Design & Software Architecture | 51 |
| 3 Structural Elements of the NLCode | 53 |

| | | |
|----------|---|------------|
| 3.1 | Arced Circular Frame | 53 |
| 3.2 | Geometry of the Digital NLCode | 61 |
| 3.2.1 | Variable Transformation: System to Plot Coordinates | 62 |
| 3.2.2 | Variable Transformation: Plot to Pixel Coordinates | 67 |
| 3.3 | Nonlinear Pattern: The Lorenz System | 75 |
| 3.3.1 | Numerical Solution | 76 |
| 3.3.2 | Variable Transformation | 77 |
| 3.3.3 | Dissipation | 79 |
| 3.3.4 | Equilibrium Solutions and Local Stability | 79 |
| 3.3.5 | Lyapunov Spectrum & Lyapunov Dimension | 81 |
| 3.3.6 | Basin of Attraction | 82 |
| 3.3.7 | Homogeneous Driving & Synchronisation | 83 |
| 3.4 | Colour Profile of the NLCode | 86 |
| 4 | The NLCode Generator | 93 |
| 4.1 | Objectives of the NLCode Generator | 93 |
| 4.2 | Spatial Overlaps | 96 |
| 4.2.1 | Initial Conditions & Transient Intervals | 97 |
| 4.2.2 | Sampling of the 2D Projected Trajectory | 98 |
| 4.2.3 | General Method: Squares & Occupancies | 99 |
| 4.2.4 | Acceptance/Rejection Criteria | 111 |
| 4.3 | Colour Overlaps – Input Palette | 114 |
| 4.4 | Bridging Worlds: NLCode’s Operating Window | 120 |
| | Part II Results & Discussion | 131 |
| 5 | The NLCode Reader via a Demo App | 133 |
| 5.1 | NLCode Capture: Target Area | 133 |
| 5.2 | Image Processing | 139 |
| 5.2.1 | Masking | 139 |
| 5.2.2 | Colour Detection & Filtering | 143 |
| 5.2.3 | Corner Detection & Identification | 166 |
| 5.2.4 | Perspective Correction | 170 |

| | |
|--------------------------------|-----|
| 5.3 NLCode Reading | 173 |
| 6 Summary & Conclusion | 177 |
| Appendix: The Dequan Li System | 178 |
| Bibliography | 185 |

Acknowledgements

I would like to give my warmest thanks to my academic supervisor Dr. Eifion Jewell for his continued support and encouragement. There have definitely been some rough times in the duration of this project, and many obstacles to overcome. I am grateful to him for maintaining a focused and solution oriented approach at all times. The lessons I learned while working with him are countless and will certainly prove most valuable in all my future endeavours. Thank you.

I am also grateful to my industrial supervisor Mr. Richard O'Neill, first of all for giving me the opportunity to work on this project, but also for having faith in me, keeping an open mind, and for his continuous encouragement. Our collaboration has certainly broaden my horizons as well.

Special thanks to the Materials & Manufacturing Academy (M2A) team for always doing their best to accommodate the students' needs. Their attentiveness allows us to better focus on our research.

I would also like to thank the sponsors of this project, Tinmasters, Swansea University, the European Social Fund (ESF) through the Welsh Government and the Engineering and Physical Sciences Research Council (EPSRC), for making this work possible.

It would be amiss of me not to acknowledge and extend my thanks to the examiners of my Thesis, for reading closely and appreciating my efforts to overcome the challenges related to this project and deliver a quality report of my findings.

Last, but certainly not least, I must express my gratitude to my teacher, Professor George Voyatzis. He has taught me all I know about Nonlinear Dynamics, and even though I cannot claim to have delved deeper into it – during my later studies – than he might have had me do, I think he might condone my taking this interesting tangential to the subject. A simple thank you will never be enough for the person who taught, encouraged and supported me through all my academic studies, but here we are Mr Voyatzi. Thank you.

List of Figures

| | | |
|------|--|-----|
| 1.1 | Market share of various industries relevant to this study | 5 |
| 1.2 | Evaluation of the anti-counterfeiting technologies considered | 13 |
| 1.3 | Workflow of an anti-counterfeiting system utilising smartphones | 16 |
| 1.4 | Proposed feature: The NLCode | 18 |
| 2.1 | Lorenz attractor | 27 |
| 2.2 | Deformation of an infinitesimal neighbourhood in state space | 28 |
| 2.3 | Geometric interpretation of the Lyapunov spectrum | 39 |
| 3.1 | NLCode's arced circular frame | 53 |
| 3.2 | Detailed view of the arced circular frame's structure | 54 |
| 3.3 | Plot-to-pixel coordinate transformation based on NLCode's frame | 68 |
| 3.4 | Lorenz attractor in 3D, before the transformation | 76 |
| 3.5 | Lorenz attractor in 3D and 2D projections, after transformation | 78 |
| 3.6 | Basin of attraction of the Lorenz attractor | 82 |
| 3.7 | Synchronisation of (x^1, x^3) response, with x^2 drive | 84 |
| 3.8 | Synchronisation of (x^2, x^3) response, with x^1 drive | 85 |
| 3.9 | A sample periodic sequence of the NLCode's colour profile | 87 |
| 3.10 | NLCode's colour profile: Sawtooth function | 88 |
| 3.11 | Colour bases of the NLCode's colour profile | 89 |
| 3.12 | Effect of the period of the NLCode's colour profile on the feature | 90 |
| 4.1 | Patterns with poor area coverage and high density in state space | 94 |
| 4.2 | The four main tasks of the NLCode Generator | 95 |
| 4.3 | The effect of transient intervals on the NLCode pattern | 98 |
| 4.4 | Sampling of the 2D projected trajectory | 99 |
| 4.5 | Visual demonstration of the method identifying Spatial Overlaps | 101 |
| 4.6 | Line width Vs square size in the Squares & Occupancies method | 102 |
| 4.7 | Partial derivation of a variable maximum sampling distance | 104 |
| 4.8 | Definition of a constant maximum sampling distance | 107 |
| 4.9 | A detailed view of the inner workings of Squares & Occupancies | 109 |
| 4.10 | Sampled trajectory treated for overlapping initial conditions | 110 |
| 4.11 | Spatial overlaps: From complete rejection to pattern acceptance | 113 |
| 4.12 | Colour overlaps: The last fort against spatial overlaps | 117 |
| 4.13 | NLCode pattern treated for yellow ending | 119 |

| | | |
|------|--|-----|
| 4.14 | Operating window: Print size Vs Working distance | 128 |
| 4.15 | Operating window: Working distance Vs Print size | 130 |
| 5.1 | Demo App – Camera View with Target Area On | 133 |
| 5.2 | Demo App – Initial Photo View & Masking prompt | 140 |
| 5.3 | Masking of the NLCode’s captured image | 143 |
| 5.4 | NLCode’s Test Photographic Sample 1 | 145 |
| 5.5 | NLCode’s Test Photographic Sample 2 | 145 |
| 5.6 | NLCode’s Test Photographic Sample 3 | 145 |
| 5.7 | Test photographic sample after cropping | 147 |
| 5.8 | Test photographic sample after conversion to HSB | 148 |
| 5.9 | Smoothed density functions of the test sample | 149 |
| 5.10 | Application of the brightness filter F_{HSB} (4-colour) | 151 |
| 5.11 | List of dominant colours in the background of the test sample . . . | 152 |
| 5.12 | Application of the F_{RGB} filter for background removal (2-colour) . | 153 |
| 5.13 | Application of F_{HSB} and F_{RGB} for background removal (2-colour) . | 154 |
| 5.14 | Application of both F_{HSB} and F_{RGB} filters for background removal | 156 |
| 5.15 | Visual demonstration of the output palette’s properties | 159 |
| 5.16 | Application of the final output palette on the test sample | 160 |
| 5.17 | Basic components of the NLCode’s colour profile | 162 |
| 5.18 | Demo App – Masked Photo View & Colour Filter prompt | 163 |
| 5.19 | Colour detection on the NLCode’s masked components | 165 |
| 5.20 | Demo App – Filtered Photo View & Corner Detection prompt | 166 |
| 5.21 | Corner detection on the NLCode’s arced circular frame | 168 |
| 5.22 | Demo App – Detected Corners View & Perspective Correction prompt | 170 |
| 5.23 | Perspective correction on the NLCode’s images | 172 |
| 5.24 | Demo App – Perspective Corrected View & Forward Tracing prompt | 173 |
| 5.25 | Demo App – Forward & Backward Tracing Views | 176 |
| A.1 | Dequan Li attractor in 3D, before the transformation | 180 |
| A.2 | Dequan Li attractor in 3D and 2D projections, after transformation | 181 |
| A.3 | Basin of attraction of the Dequan Li attractor | 185 |

List of Tables

| | | |
|-----|--|-----|
| 1.1 | Extended classification of anti-counterfeiting technologies | 8 |
| 1.2 | Anti-counterfeiting technologies used in security printing | 12 |
| 1.3 | Anti-counterfeiting features utilising smartphone technology | 17 |
| 2.1 | Stability of equilibrium solutions in 3D dynamical systems | 35 |
| 2.2 | Types of attractors in 3D dissipative systems | 47 |
| 3.1 | Specification of the arced circular frame of the NLCode | 60 |
| 3.2 | Stability of equilibrium solutions in the Lorenz system | 81 |
| 4.1 | Primary camera specifications of the iPhone 6S Plus | 122 |
| 5.1 | The brightness filter F_{HSB} | 150 |
| A.1 | Stability of equilibrium solutions in the Dequan Li system | 184 |

Acronyms & Abbreviations

| | | |
|-------|---|---------------|
| 1D | one-dimensional | 3 |
| 2D | two-dimensional | 2, 32 |
| 3D | three-dimensional | 28 |
| CO | Colour Overlaps | 100, 119 |
| CT | Continuous-time | 28 |
| DT | Discrete-time | 28 |
| EU | European Union | 6 |
| EUIPO | European Union Intellectual Property Office | 6 |
| IC | Initial Condition | 29 |
| IEC | International Electrotechnical Commission | 153 |
| iOS | iPhone Operating System | 139 |
| ISO | International Organization for Standardization | 9 |
| LP | Logical Point | 142 |
| MP | megapixel | 19 |
| NFC | Near Field Communication | 14 |
| ODE | Ordinary Differential Equation | 28 |
| OECD | Organisation for Economic Cooperation and Development | 6 |
| OW | Operating Window | 125, 133, 135 |
| RFID | Radio Frequency Identification | 13 |
| RK4 | 4th-order Runge-Kutta | 29 |
| S&Os | Squares & Occupancies | 104 |
| SO | Spatial Overlaps | 100, 101 |
| SoC | System on a Chip | 19 |
| SRF | Smallest Resolvable Feature | 127 |
| UI | User Interface | 140 |

- UIB** User Interface Button [141](#)
- UIE** User Interface Element [140](#)

1

Introduction

The project whose initial phases, definitive decisions, and gradual progress up to the final results are presented in this thesis, was an initiative of Tinmasters' Chief Executive Officer (CEO), Mr. Richard O'Neill. Given the widespread of counterfeit products, managing a major supplier of steel packaging to premium brands in various parts of the world, raises the demand for added security features that offer user-based product authentication. In order to keep Tinmasters on top of the newest developments in the anti-counterfeit packaging industry, Mr. O'Neill established a continuing collaborative relationship with Swansea University. This project stemmed from this collaboration.

The amazing opportunity offered by this project to a physicist with computational tendencies – about to dive into an engineer's world, has always been the project's openness. *"The development of an anti-counterfeiting technology from first principles"*, or something along those lines, is indeed a project title for the creative theorist with a handy toolset.

As one may have expected, this project begun somewhere in a wide spectrum of options and blossomed into a multidisciplinary topic. This introductory chapter focuses on providing the reader with the broader scope of this project, and a clear view of the choices involved in the narrowing of that scope and the creation of the coding scheme presented.

[Section 1.1](#) lists the objectives of this thesis from an academic perspective. As the reader will come to realise by this end of this chapter, the proposed scheme met most of its industry-oriented requirements from the early drafts of its design, if not upon conception; its academic requirements are what fills the gaps between *"ticked boxes"*, so to speak. The socioeconomic background relating to counterfeit and pirated goods that motivated this project provides context, and is briefly surveyed in [Sec. 1.2](#). The project's objectives from an industrial perspective formed the criteria based on which an initial set of anti-counterfeiting technologies was gathered, grouped, and evaluated. This initial phase of the project is what one might call *Literature Review*, and is presented in [Sec. 1.3](#). The industrial objectives are listed within that section ([Subsec. 1.3.1](#)), and the proposed scheme is given a brief overview in [Sec. 1.4](#).

1.1 Thesis Objectives & Main Focus

The research presented in this thesis was initiated with the purpose to provide the current landscape of brand protection, with an alternative security feature. In order to make a meaningful contribution to any field, the first two objectives should always be to

- Understand and place the problem into context, and
- Thoroughly review the existing solutions, or attempts to a solution.

The *problem*, in the present context, is *counterfeiting*, and the *solution* includes a wide variety of *anti-counterfeiting technologies*. The *context*, which helps narrow down the available options and/or alternative courses of action, is provided by identifying

- The source of the problem, and
- The application area of a potential solution.

The *source* of the problem in the case presented is not simply counterfeiters, but maybe more importantly the *operational capabilities of counterfeiters*, which underlie and are reflected upon the spread of the counterfeit market. The *application area* identifies with the project initiator (Tinmasters), and is one of the main application areas of anti-counterfeiting technologies, that is, the *packaging* of products. A thorough Literature Review addresses the above matters and helps

- Define the precise field of research.

Soon after the completion of the Literature Review, the project took a definitive direction toward the specific *field* of *printable two-dimensional (2D) codes that use chaotic trajectories, which can be retrieved by means of synchronisation, using a smartphone camera*. Placing the focus of this research on the development of a coding scheme adhering to the above specifications, defined the following set of objectives and subordinate tasks:

- **Design:** At the highest level, the two main components of the proposed scheme are a *code generator*, and a *code reader*. Firstly, since the *code* must be *printable* and *aesthetically pleasing*, it has to be 2D and form an *interestingly complex pattern*, possibly coloured by way of visual appeal. Chaotic attractors

do possess the latter characteristic, but in 2D, the underlying dynamical systems are *discrete* (Chapter 2). *Continuous* dynamical systems only exhibit chaotic attractors in *three or more dimensions* (Sec. 2.1, footnote 9, and Sec. 2.6). Suppose that the above remarks determine that the code generator is to create trajectories evolving within a chaotic attractor; in 2D the code would utilise both of the trajectory's components, and in 3D just two of them. Secondly, the code must be readable based on a photograph of its printed copy. This means that any *code generator*, must be accompanied by, and enable, an appropriately chosen *code reader*, capable to retrieve the trajectory.

The seemingly “casual” mention in the above description, of terms such as *continuous/discrete dynamical system*, and *chaotic attractor*, is not accidental; it aims at defining the main areas that needs to be included in the background research of this project:

- ***Nonlinear Dynamics & Chaos*** (3D continuous dynamical systems)

Image processing was not mentioned above, not because it is not an integral part of the scheme's design and operation, but because it also involves the second main element of the feature, which has not yet been mentioned. Getting from a “pretty” line drawn, printed, and subsequently captured by a smartphone camera, to extracting a sequence of numbers forming that line, requires a series of preprocessing steps that include *colour/feature detection*, *alignment*, *perspective correction*, and *coordinate transformations*. Most of those steps heavily rely on image processing techniques, which are enabled by a frame especially designed for this purpose. This frame encloses the chaotic trajectory and defines the feature's bounds. Most of the image processing techniques used in this project are preexisting and well known, but some of them received appropriate treatment in order to accommodate the needs of the developed scheme. Their application comes into play later in the design process – borderlining with the next central objective of the thesis, which is *implementation*, but constitute a main topic of research nonetheless:

- ***Image Processing*** (masking, filtering, feature extraction, warping)
- **Implementation:** As challenging as the design of any kind of system may be, as thorough as the background research necessary to bring insightfulness into it must be, ultimately, the design itself simply provides the prescription according to which the system is to be created. The implementation of each individual

system component identified in the design, as well as the link between those components, is one of this thesis' main objectives, and is mainly computational.

- **Report:** Reporting – or otherwise publicising – one's findings is an inextricable part of any type of research. This is how any kind of progress works; by building upon the work of others instead of keeping busy reinventing the wheel.

1.2 The Counterfeiting Problem

The term *counterfeiting* refers to the intentional alteration of a product itself, its packaging, or any identifying and/or certifying information related to it, with the malicious intent of illicit trade (National Electrical Manufacturers Association (NEMA), 2009). In recent years, the underground market of counterfeit products has grown into a global network that poses an increasing threat to modern societies on all levels. The diversity of the product types targeted by counterfeiters – ranging from automotive parts, computer hardware and mobile phones to pharmaceuticals, packaged foods/beverages, and tobacco products – turned what has since become known as the *counterfeiting problem* into a major concern of increasing priority for the general public, brand owners, and governments.

According to the latest – 2021 – study conducted jointly by the [Organisation for Economic Cooperation and Development \(OECD\)](#) and the [European Union Intellectual Property Office \(EUIPO\)](#), in 2019, the counterfeit and pirated products comprised 2.5% of the world trade (equivalent to US\$464 billion), and 5.8% (US\$134 billion) of the products imported in the [European Union \(EU\)](#) (OECD/EUIPO, 2021). Earlier analyses performed by OECD and EUIPO showed that in 2016, counterfeit and pirated goods amounted to 3.3% of the international trade (equivalent to US\$509 billion in nominal terms) and 6.8% (US\$134 billion) of EU imports (OECD/EUIPO, 2019), while the respective estimates based on data from 2013 are 2.5% (US\$461 billion) and 5% (US\$116 billion) (OECD/EUIPO, 2016). These results indicate that in the recent 6-year period from 2013 to 2019, the counterfeit market has only been subject to insignificant variations which left it virtually unaltered and as dangerous as it has been since its accelerated rise during the first decade of the century.

Figure 1.1 shows the market share of the world trade, the total international trade in counterfeit and pirated goods, the anti-counterfeit packaging, the metal packaging, and the market of anti-counterfeit technologies, over a span of several years. The plot shown was created using data collected from the websites of ten global market research and analysis companies¹. These companies issue analysis reports on various markets which typically cost a few thousand US dollars each. The data used to make this plot were gathered from the report overviews included in the companies' websites for promotional purposes. The kind of analysis performed by each of those teams set their original requirements in input data, and determined their forecast period – a central feature in these kinds of reports. The effect of this is evident in the plot where each plot-line starts and ends at different years from the rest. Note that, due to the wide range of market share values, the corresponding axis is displayed in logarithmic scales.

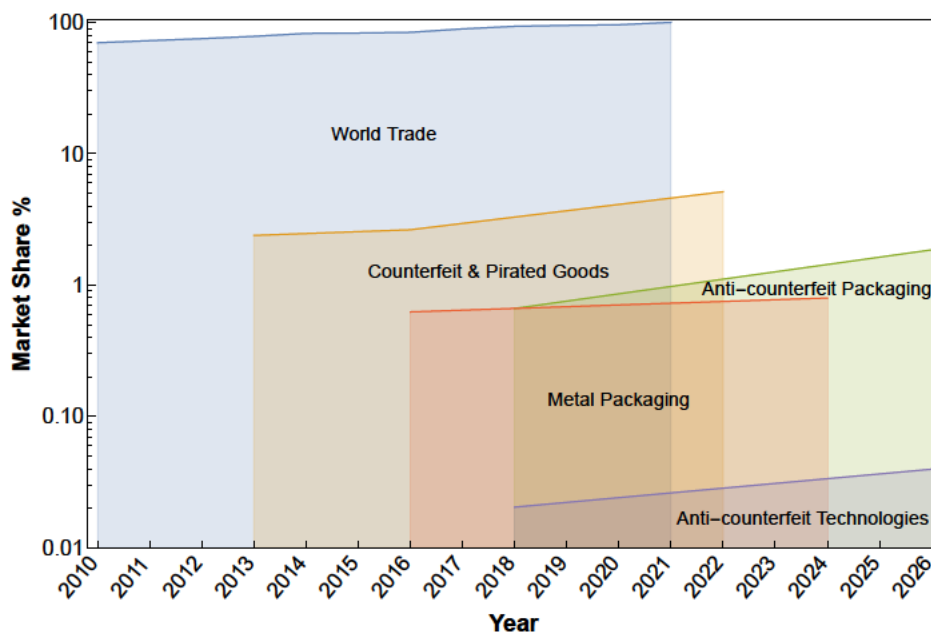


Figure 1.1: Market share of various industries relevant to this study, against world trade. The market values obtained from the sources (footnote 1) in US\$, were converted to market shares by setting the highest value of the world trade – US\$19.3 trillion in 2021 – to 100%, and then calculating the rest using that as a point of reference. The qualitative picture drawn by this graph is that the markets dedicated to combating the trade of counterfeit products are expected to grow at a faster pace than those unrelated to anti-counterfeiting.

¹(Credence Research, 2018; Goldstein Research, 2019; Market Research Future, 2019; MarketsandMarkets, 2021; Mordor Intelligence, 2020; Orbis Research, 2019; Reports & Data, 2019; Research Nester, 2022; Smithers, 2019; Verified Market Research, 2018)

Finally, it is important to stress the fact that this plot is provided merely as an *indication* that the trend of rapid growth established in the OECD/EUIPO reports mentioned earlier, seems to be reflected on the global market of security packaging and anti-counterfeit technologies, both of which appear to exhibit a similar trend and a projected increase up to year 2026.

The most serious danger posed by counterfeit products is to public health and safety. Counterfeit medicines, foods and beverages, or even clothes, toys and home electrical appliances that may contain hazardous chemical substances or substandard components, enter the legal market unchecked in terms of compliance to safety regulations. As a consequence, consumers using such inferior products are exposed to severe or even life threatening conditions. Jeopardized brand integrity causes manufacturers and brand owners to face liability issues and major loss of revenues. The decrease in sales of legitimate products caused by the counterfeit markets leads to increased prices on one hand, and higher unemployment rates on the other. At the same time, governments suffer losses from tax revenue while being forced to increase the law enforcement measures to combat counterfeit markets.

Recognizing the need to offer manufacturers the tools necessary to protect their brands and more importantly their clients and general public from counterfeiting, the [International Organization for Standardization \(ISO\)](#) issued in 2012 a standard, specifying performance criteria for the technical solutions adopted by companies as a means to validate the authenticity of their products, as well as an evaluation methodology for these technologies. According to ISO's website², ISO 12931:2012 - *Performance criteria for authentication solutions used to combat counterfeiting of material goods* (ISO, 2012), has since been revised and the updated version was published in 2020 as ISO 22383:2020 - *Security and resilience – Authenticity, integrity and trust for products and documents – Guidelines for the selection and performance evaluation of authentication solutions for material goods* (ISO, 2020). The essence in both versions remains the same; these standards are intended to act as a guide to industries, which are only required to find suitable anti-counterfeiting technologies that meet the performance criteria specified, but are in no way restricted as to how their selected approaches will achieve that. For this reason, the performance criteria are provided in the form of

²Link to ISO's web page containing information about ISO 12931:2012 and ISO 22383:2020: <https://www.iso.org/standard/52210.html>

general guidelines for any organisation to develop and apply an effective crime prevention strategy against product fraud. The development process proposed includes risk assessment, identification of the intentions, motives and opportunities for product fraud, a classification of product fraud activities and fraudsters' profiles, the selection and implementation of suitable countermeasures, and finally an assessment of the overall effectiveness of the approach adopted.

Tinmasters, the industrial sponsor of this project, specialise in the commercial printing and coating of tinplate and aluminium substrates for the metal packaging industry. As a company with a well-established position among the leading industries in the field since 1909, it has always been highly invested in keeping up to date with the newest developments and maintaining the highest standards of excellence. Tinmasters have been continuously commissioned by infant formula manufacturers since 1978. The types of services they offer, along with the rapid rise of the counterfeit market which culminated in the 2008 Chinese milk scandal (Wikipedia Contributors, 2023a, and references therein), placed the company at the centre of the aforementioned developments. Tinmasters took on this challenge by initiating the current project with the aim to first identify the anti-counterfeiting technologies that best suit their manufacturing process, and subsequently develop and provide their clients with an efficient brand protection scheme.

1.3 Overview & Evaluation of Anti-Counterfeiting Technologies

Any countermeasure against counterfeiting meant to be incorporated into a product or the product's packaging and add some form of technology-based authentication capability is called *anti-counterfeiting technology*. If the classification given in the ISO 22383:2020 standard is to be taken as "*the standard*", then anti-counterfeiting technologies are divided into three broad categories: *Overt*, *covert* and *forensic technologies* (Spink, 2012, and references therein). Specifically,

- Overt technologies are implemented in security features which are readily visible, or more precisely *detectable* and *verifiable* without the use of specialised equipment such as the wide variety of proprietary scanning devices of the past, and smartphones and smartphone applications of the present.

Table 1.1: *Extended classification of anti-counterfeiting technologies. Manufacturers of security features and market analyst firms go beyond ISO's coarser classification into overt, covert and forensic technologies, by introducing the semi-overt and semi-covert types, and eliminating the forensic class, whose technologies are absorbed by the general class of covert technologies. The left column gives an all inclusive list of the proposed types of anti-counterfeiting technologies, and the right column provides representative examples of each category (L. Li, 2013; National Electrical Manufacturers Association (NEMA), 2009).*

| Type | Anti-counterfeiting Technology |
|---|---|
| <p style="text-align: center;">Overt <i>Readily visible</i></p> | <p>Tamper evident closures & labels (shrink sleeves, tear tapes, delaminating/destruct films) Security substrates (papers, threads) Watermarks Optically variable films & inks (floating/sinking, colour shifting) Pearlescent & metallic inks Intaglio printing Guilloché patterns Holograms Product serialisation</p> |
| <p style="text-align: center;">Semi-overt <i>Inconspicuous</i></p> | <p>Fluorescent & Phosphorescent inks Photochromic & thermochromic inks Metameric inks Reactive inks</p> |
| <p style="text-align: center;">Semi-covert <i>Machine readable</i></p> | <p>Barcodes Matrix codes (2D, 3D) Electronic Product Code Radio Frequency Identification tags Near Field Communication tags</p> |
| <p style="text-align: center;">Covert <i>Invisible</i></p> | <p>Infrared & Ultraviolet inks Halftone dots designs Micro/nano printing</p> |
| <p style="text-align: center;">Forensic <i>Lab equipment</i></p> | <p>Taggants (biological, chemical, micro/nano-technology) Quantum Dots</p> |

- Covert technologies include all those features that are
 - Inconspicuous, in the sense that one has to be aware of their existence in order to notice them,
 - Detectable but not verifiable without specialised devices, or
 - Completely invisible, in which case both the detection and the verification of the feature rely on specialised equipment.
- Forensic technologies are essentially covert technologies of the third of the above types; their detection and verification however, can only be performed in laboratories by specialised personnel.

Adding to the above classification, every manufacturer of security features and any market research company like the ones cited earlier, classifies anti-counterfeiting technologies based on more refined sets of criteria. This results to the addition of one or two categories between the overt and covert classes, named *semi-overt* and *semi-covert* technologies, and the elimination of the forensic category in favour of the class of covert technologies. [Table 1.1](#) groups several anti-counterfeiting technologies based on an extended classification that includes all proposed categories.

1.3.1 Project Objectives from an Industrial Perspective

The project presented in this thesis is titled “*Brand Protection for Steel Packaging Products*”. The two main tasks during its phase of initiation were first, a comprehensive overview of the technologies forming the current landscape of the *anti-counterfeit packaging* market in general and, pertinent to Tinmasters’ specialty, the *metal packaging* market in particular. The second task was the development of a scheme to evaluate a selection of those anti-counterfeiting technologies that – for one reason or another – stood out during the review process.

The objectives of this project from an industrial perspective, as they translate into general guidelines for the first part of the research, were the following:

1. To strive for the development of a *novel* technology from first principles.
2. To place *emphasis on overt technologies*, since apart from being objectively safe, a product must also be perceived as safe.

3. To aim for the development of a technology with *high aesthetic appeal*, targeting premium brands not only through their investment in high quality package designs, but also because an aesthetically pleasing feature creates an image of sophistication and reliability.
4. To nonetheless proceed allowing the possibility for certain *covert* and/or *forensic technologies* to serve as *additional layers of security*, moving toward a very well known approach referred to as *layering* (National Electrical Manufacturers Association (NEMA), 2009). Layering is based on one of the fundamental principles often applied in the development of security systems, which states that *no single technology will ever be able to provide absolute protection against fraud*. It aims at finding suitable ways to combine multiple technologies, each providing different levels of protection. In the context of anti-counterfeiting, this approach can help manufacturers and brand owners address multiple objectives with a single solution (van Renesse, 1998).
5. To carefully consider and, if possible, try to minimise the impact of implementing the developed technology on the manufacturing process. Without it meant to be restrictive, this guideline immediately places emphasis on security features that can be *printed on the metal substrates* Tinamasters work with.


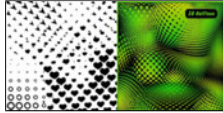
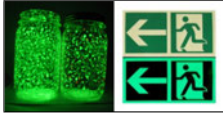

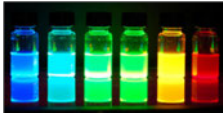


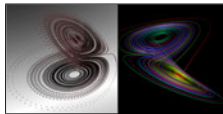








1.3.2 Candidate Technologies & a Scheme for Their Evaluation

The technologies reviewed during the initial phase of this project are the result of years of scientific research and technological advancements. Some of them were implemented as security features at an industrial scale using well established and standardised manufacturing processes 30 years ago, while others involve more recently developed techniques and processes. These technologies are specifically used in *security printing* and are divided in two categories. The first category is called *security inks* and includes *fluorescent, phosphorescent and quantum dot pigments, metameric inks, pearlescent and metallic inks*, as well as *photochromic and thermochromic inks*. The second one, called *security patterns*, includes technologies such as *halftone dots design, micro and nano printing, multi-parametric fine-line patterns*, e.g. *guilloché*, and *“strange” or “chaotic” patterns, 2D matrix codes, holograms*, and *printed electronics* including *Radio Frequency Identification (RFID)* and *Near Field Communication (NFC) tags* (Nathe, 2012). Table 1.2 provides a brief overview of these technologies.

The rating scheme developed for the evaluation of the technologies considered consists of the criteria listed below. Each of the technologies listed in [Table 1.2](#) was rated on a scale from 1 to 5 according to the selected criteria. The results of the evaluation process are presented in the *discrete density plot* of [Fig. 1.2](#). Since other factors, such as experience and intuition also play an important role in the evaluation process, this scheme is only meant to serve as a guide toward the identification of the technologies with the highest potential to offer an effective solution.

1. *Maturity*: The maturity of a technology refers to the extent, scale, and types of products this technology has been applied to in the past. A mature technology offers the advantage of a standardised method of implementation. This same advantage, however, becomes a disadvantage if one considers the ease with which such a technology can be copied or otherwise manipulated by an adversary.
2. *Security*: The security a technology provides is evaluated based on whether it is overt, covert, or forensic, and whether it has the ability to store information or not. Its level of security also depends on the type of authentication it offers, and the stage of the supply chain it is addressed to.
3. *Food Safety/Contact*: Some of the existing technologies use materials which have either be proven to be toxic, or are perceived as hazardous. Considering that Tinmasters offer their services to the foods and beverages packaging industry, this criterion evaluates a technology based on its compliance to health and safety regulations, and is applied with high sensitivity.
4. *Aesthetics*: With an enhanced capability to positively predispose potential buyers, an aesthetically appealing security feature (also see [Subsec. 1.3.1](#)) becomes a great fit for premium brands and their products, adding value to the services Tinmasters offer as a result.
5. *Compatibility*: Each of the technologies considered is also evaluated in terms of its ease of implementation. A highly favoured technology is expected to be fully compatible with, or at least have minimal impact on Tinmasters' manufacturing process.
6. *Cost Efficiency*: A technology is initially assessed in terms of production costs. Its final rating according to this criterion attempts to reflect the trade-off between the technology's cost and its estimated added value.

Table 1.2: Anti-counterfeiting technologies used in security printing.

| SECURITY INKS | | SECURITY PATTERNS | |
|---|--|--|--|
| <p>Fluorescent</p>  | Invisible/white under visible light displays another color under UV/IR light | <p>Halftone Dots</p>  | Customized dot shapes or patterns with a recognizable design |
| <p>Phosphorescent</p>  | Invisible/white under visible light re-emits several hours after exposure | <p>Micro/Nano Print</p>  | Printed patterns or characters detectable upon magnification |
| <p>Quantum Dots</p>  | Semiconductor nanoparticles with a highly tunable, size dependent emission | <p>Guilloché Printing</p>  | High resolution interlaced patterns of lines with variable interline distances |
| <p>Metameric</p>  | Pairs of inks whose colour contrast is visible only under a certain light source | <p>"Strange" Patterns</p>  | Complex patterns produced by non-linear dynamical systems |
| <p>Pearlescent</p>  | Layered, flake-like particles interact with light and create an effect of luster | <p>2D Matrix Codes</p>  | Information storing patterns of uniquely configured cells with contrasting colours |
| <p>Metallic</p>  | Specular and diffuse reflection of light on particles creates a metallic sheen | <p>Holograms</p>  | Interference patterns from a scene act as diffraction gratings to recreate it |
| <p>Photochromic</p>  | Disappears when exposed to artificial light other than natural or UV light | <p>RFID</p>  | Printed electronic devices (tags), store information to track and trace products |
| <p>Thermochromic</p>  | Exhibit colour-changing effects controlled by temperature | <p>NFC</p>  | Similar to RFID tags, these devices operate at different (shorter) distances |

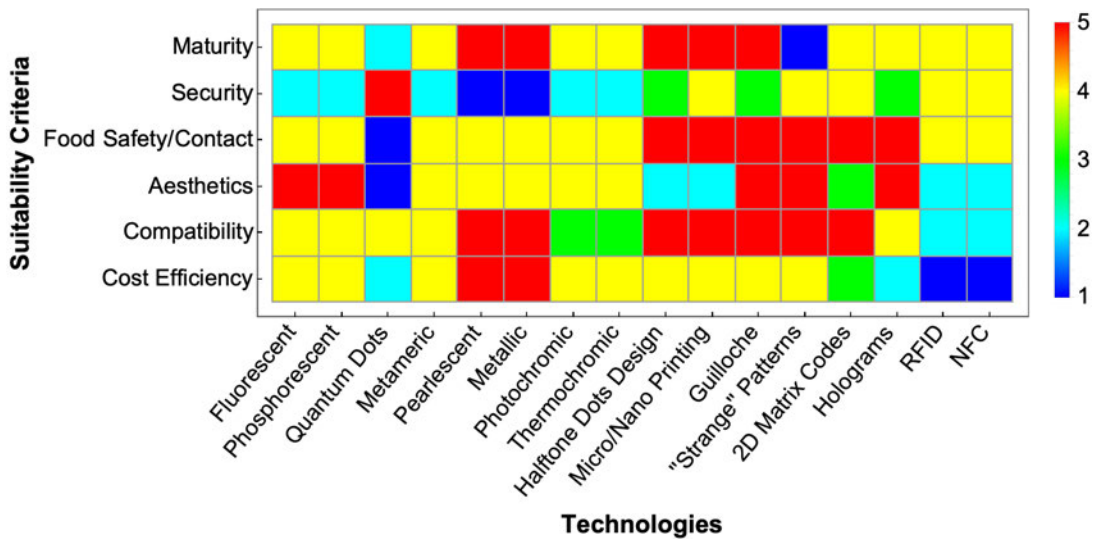


Figure 1.2: Evaluation scheme for the anti-counterfeiting technologies considered. Every technology is rated on a 1 – 5 scale, in terms of its performance on each of the suitability criteria set. With the exception of the criterion related to maturity, which can be interpreted as having both a positive and a negative impact on the suitability of a technology, the rest of the criteria are defined in a way that a higher rating indicates better performance.

In the beginning of this subsection, while introducing the two types of technologies used in security printing, the technology related to some “*strange*” or “*chaotic*” patterns was mentioned in passing, without any explanation as to what this technology is. The author assumes that, having read [Sec. 1.1](#), which introduces the main objectives of this thesis, the reader has become ever so slightly familiar with the coding scheme presented in this thesis, whose formal introduction will be given in [Sec. 1.4](#). The reason it was made a part of the evaluation process, is that it seemed interesting to see how it compares against the other candidate technologies, and maybe partially justify its prevalence over them.

Judging by the dominance of the warm colours in general, and yellow in particular, the colour-coded density plot of [Fig. 1.2](#) suggests that most technologies fared quite well against the criteria they were tested on. It also indicates, however, that the selection process that preceded the evaluation, created a high quality sample of candidate technologies to begin with. Without delving too much into details, the technology or technologies with the highest potential should have the highest ranking across the list of criteria which, in terms of [Fig. 1.2](#), it translates to columns dominated by red. At first glance, the technology with the highest

score is immediately seen to be the guilloché pattern, with the highest score – 5 – in 4 out of 6 criteria. It is followed by “strange” patterns, halftone dot designs, and micro/nano printing, all of which scored a 5 in 3 out of 6 criteria.

However, as already noted earlier, the maturity of an anti-counterfeiting technology is a property that cuts both ways. This starts to become obvious when noticing that maturity is the criterion in which the guilloché pattern received one of its highest scores. Indeed, guilloché patterns have been used against counterfeiting since the middle of the 19th century (De Leeuw & Bergstra, 2007). It is therefore fair to say that adversaries have had plenty of time to hone their skill at faithfully reproducing these fine-line patterns. “Strange” patterns, on the other hand, are completely new and therefore only as mature as the technologies surrounding their implementation, i.e. printing and photographic capturing. The fact that the implementation of 2D matrix codes – a technology that is considered quite mature – has exactly the same dependencies, suggests that a viable design can easily push this new technology to first place, a claim that is strongly complemented by the observation that with their potential to store information, “strange” patterns match 2D matrix codes in security and completely outrank the guilloché patterns.

1.3.3 A Quick Look at Recent Trends

Anti-counterfeiting technologies that rely on the fast-growing capabilities of smartphone devices in order to provide user-based product authentication, started becoming the favoured approach to the battle against counterfeiting a little more than 10 years ago. That was when smartphones began featuring high-resolution cameras (Wikipedia Contributors, 2023b, and references therein). Today, smartphones implement several more advanced technologies that allow them to perform a series of highly sophisticated tasks. Pairing only a few of the fundamental technologies in a smartphone’s “arsenal” with the tasks facilitated by each of them, enables smartphone applications that implement anti-counterfeiting features, to pack instructions for data collection, processing and data exchange, as well as the final decision regarding the authenticity of a product. The three main pairings between tasks and technologies are the following (Baldini & Pons, 2017):

■ *Data acquisition:*

- High-resolution cameras – above 5 *megapixel (MP)* – enable smartphones to capture the details necessary for the adequate processing of certain optical security features.
- RFID/NFC functionality enables smartphones to authenticate products carrying RFID and NFC tags.

■ *Data processing:*

- High-performance *Systems on a Chip (SoCs)* equip smartphones with advanced processing capabilities and enable them to carry through the computationally intense calculations often required by anti-counterfeiting applications.

■ *Data exchange:*

- Wireless connectivity of a fast-growing capacity gives instant download access to the dedicated application of any anti-counterfeiting feature a user may come across. More importantly, however, it establishes a line of communication with a remote server for *data matching*, that is, the comparison of the acquired data against the data hosted in a remote database.

A typical anti-counterfeiting system utilising smartphone technology for product authentication includes all the functional elements listed above and has the workflow schematically represented in [Fig. 1.3](#). In the context of this operation scheme, the *brand owner* and the potential customer, or *consumer*, can be respectively referred to as the *client* – of the company offering the anti-counterfeiting solution, and the *end-user* – of the anti-counterfeiting smartphone application or *app*, for short, and ultimately of the product. Assuming, as an example, that the anti-counterfeiting feature works by somehow encoding a serialisation number at the batch level³, once the client issues the serial number identifying a particular batch of products, two processes get initiated: The *batch number* is (i) used to initialise the *generator* of the anti-counterfeiting feature – a computer program or other device that embeds the identifying information (serial number) into the feature, and (ii) registered in the remote *database server* for later use in data matching. After this stage, the database server enters a standby mode while the

³Of course, instead of a number it can be any other identifying information, as long as there exist appropriate mechanisms by which this information can be embed into the feature and later be retrieved from it.

generator passes onto the *application* process, the instructions for the incorporation of the feature into the product. The product then gets passed along the rest of the manufacturing stages (not shown), and is distributed reaching the shelves of the retailers. The interested customer points their smartphone to the anti-counterfeiting feature which collects the necessary data and issues a request for authentication. This request is handled in two steps: The initial data gathered usually need to be processed in order to retrieve the identifying information, and the identifying information retrieved must then be sent to the remote database for data matching. Depending on the computational intensity of the initial data processing, this stage can be performed on site, i.e. by the smartphone, or the data can be sent to the remote server for processing. Either way, the final decision regarding the authenticity of the product is issued by the server and sent back to the smartphone which notifies the end-user accordingly.




Figure 1.3: Workflow of a typical anti-counterfeiting system utilising smartphone technology for product authentication: The brand owner issues a serialisation number that initialises the security feature generator, and gets registered in to remote database. From this point onward the process follows the top branch first to the incorporation of the feature into the product or its packaging, and then the distribution channels until it reaches the potential buyer. The authentication process follows the bottom branch with data either being processed on the smartphone or being sent to the remote server for processing and data matching, and the final decision regarding the authenticity of the product is sent back to the smartphone which notifies the user accordingly.

The anti-counterfeiting features presented in [Table 1.3](#) are a small but representative sample of the technologies developed since 2009. These features all utilise smartphone technology for product authentication, they all implement the layered approach discussed in [Subsec. 1.3.1](#), and as systems, share more or less the workflow represented in [Fig. 1.3](#).

1.3. OVERVIEW & EVALUATION OF ANTI-COUNTERFEITING TECHNOLOGIES 17

Table 1.3: Anti-counterfeiting features utilising smartphone technology. A representative sample of the anti-counterfeiting technologies introduced since 2009, that rely on the advancements on smartphone hardware and software to offer user-based product authentication. All of these systems implement a layered approach to anti-counterfeiting and operate on a scheme similar to the one shown in Fig. 1.3.

| | |
|--|---|
| <p>Authentic Vision (2018)</p>  | <p>Track and trace security tag that provides user authentication via a smartphone app as well as real time information about the status and location of the product. It consists of three components: A “random” holographic structure (patented), a 2D barcode data matrix and human readable serialisation.</p> |
| <p>Certilogo (2017)</p>  | <p>Certilogo offers its clients a choice between the NFC Code which offers product authentication via an NFC-enabled smartphone, a QR Code which can be scanned using a smartphone camera, and their Certilogo Number Code which can be typed in a dedicated smartphone app and receive instant verification.</p> |
| <p>Confidex (2016)</p>  | <p>Hard NFC tags or printable labels that offer product identification, authentication and tracing, via an NFC enabled smartphone. Confidex has created a patent-pending RFID antenna which overcomes the performance issues caused by metallic surfaces and moisture. These tags can be programmed by a smartphone app.</p> |
| <p>Cypheme/Noise Print (2015)</p>  | <p>Cypheme’s solution is a tag that consists of a proprietary label called <i>Noise Print</i>, made using a special ink that creates a unique and copy resistant pattern when it dries. This is surrounded by several detection enabling features, all enclosed in a frame printed in a second special ink of a unique orange tint. No app installation required.</p> |
| <p>DSS/Authentisuite (2013)</p>  | <p>Document Security Systems, Inc. (DSS) offers a three-component system. The Mark is a QR code hidden using a copy resistant technology called Prism (patented). The Application is a smartphone app used to read the invisible QR code, and the Portal is a database that provides instant authentication and business intelligence.</p> |
| <p>AlpVision/Cryptoglyph (2009)</p>  | <p>Cryptoglyph is an invisible marking that embeds a pseudo-random pattern of micro-dots in the imperfections of the printed material (prepress), or a pattern of micro-holes to the coating (overprint). These patterns contain encrypted information which can be retrieved from a photograph taken by a mobile phone.</p> |

1.4 Proposed Scheme: The NLCode

The coding scheme developed can be succinctly described as “A 2D code for data representation”. Even though the terms *coding scheme*, or simply *scheme*, and *feature* have been and will be used interchangeably throughout this thesis, technically speaking, the *feature* is the *visual realisation* of the scheme, an example of which is shown in Fig. 1.4. What one sees in this figure is an overlay of two plots, that is, a graphical representation of the main components of the scheme, which after being plotted, become the main structural elements of the feature.

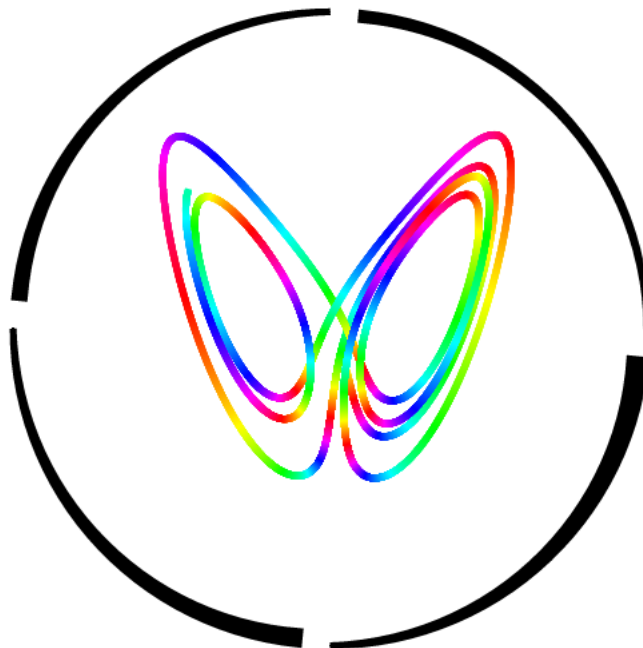


Figure 1.4: Proposed feature: The NLCode: The main structural elements of the Non-Linear Code, are the following: (a) The nonlinear pattern, which is a trajectory of a 3D, continuous dynamical system operating within its chaotic region, projected in 2D. In the example shown, that system is the well known Lorenz system (E. Lorenz, 1963). (b) The colour gradient applied on the pattern, called the colour profile of the feature. It consists of a small number of distinct colours, which are interpolated and repeated on the pattern in a periodic fashion. In the case presented, the base-colours are six: yellow, red, magenta, blue, cyan, and green. (c) The frame enclosing the pattern, called arced circular frame. It consists of a small number of arcs ending in progressively increased widths. in this case shown, the arcs are four. Each structural element of the feature corresponds to a main component of the coding scheme developed, and serves a distinct purpose (Subsec. 1.4.1).

1.4.1 Structure & Name Origin

Starting from the center of the feature and moving radially outward, the *three* main structural elements of the feature shown in Fig. 1.4 are the following:

- **Nonlinear Pattern:** This is the coloured interlaced line shown in Fig. 1.4. As already mentioned in several parts of the preceding sections, this *nonlinear pattern*, is a trajectory captured by the “*strange*” attractor of a 3D, continuous, nonlinear dynamical system displaying chaotic behaviour, projected to two dimensions. This type of system is mathematically represented by a system of three 1st order Ordinary Differential Equations, and its numerically obtained solution consists of a sequence of points in 3D, each representing a state of the system. A projection of this trajectory onto one of the three mutually perpendicular planes of a 3D Cartesian coordinate system, is obtained by simply dismissing the trajectory’s component that corresponds to the axis normal to the projection plane. The main function of the nonlinear pattern is to guide the process of its own retrieval.
- **Name Origin:** The nonlinear pattern, as well as the proposed scheme/feature as a whole, take their names from one of the main properties of the dynamical systems producing these types of trajectories, which is *nonlinearity*⁴. The name of the coding scheme developed abbreviates the hyphenated word *non-linear* as *NL*, uses it as a prefix, and appends to it the defining noun *Code* to form *NL Code*, or better yet *NLCode*, according to the author’s preference. This name is also meant to distinguish this feature from other 2D coding schemes, such as the very well known *QR Code* (ISO, 2015), the *Data Matrix* (ISO, 2006), the *Aztec Code* (ISO, 2008), Apple’s latest *App Clip Code* (Apple Inc., 2021) – to name a few, all of which represent data by contrasting light, medium(-brightness), and dark modules arranged in some *canonical order* – either along the horizontal and vertical directions, or along the paths formed by concentric squares or circles.

⁴The fact that most previous mentions of the types of systems this thesis places its focus on – and most things related to them – were termed *chaotic*, should not be taken to imply any equivalence between the latter term and *nonlinearity*. While continuous linear systems can only be chaotic in infinite dimensions, which implies that nonlinearity is a necessary condition for a system in finite dimensions to exhibit chaos, the converse is not true, that is, not all nonlinear systems are chaotic.

- **Colour Profile:** This is the colour gradient applied on the nonlinear pattern (Fig. 1.4). Even though it may seem like just a feature of the pattern added for visual appeal, it is more than that. The *colour profile* of the NLCode is a gradient created from an original set of distinct colours using an interpolation formula, after providing it with *weights* of those colours, based on a periodic function of each point's index in the pattern. Despite this being a standard technique used to create colour gradients, the period of the function assigning weights to colours is then used to endow the colour profile with a functionality which is of great value to the entire scheme. As mentioned in Sec. 1.1, in order to retrieve the trajectory, the code reader has to apply a tracing method on the nonlinear pattern. The purpose of the colour profile is to assist that method, and for this reason, its functionality is anticipated and heavily elaborated on, long before the code reader enters the workflow of the development process, that is, in the code generator. Different periods of the colour profile create different colour schemes – by altering the “*spread*” of each *base-colour*, without changing the order in which these colours appear. One of the code generator's main tasks is to identify periods that facilitate the tracing method of the code reader.
- **Arced Circular Frame:** This is the black frame surrounding the nonlinear pattern of the feature. The *arcid circular frame* consists of a small number of arcs, all of which begin with the same width, which is the line width of the nonlinear pattern. Each of the arcs ends wider than it begun, but also wider than the previous arc ended. Its two main characteristics are its black colour, which contrasts with the white background of the feature, and more importantly, the corners of the arcs. The detection of those corners, and their identification, both in terms of the arc they belong to, and the place they have on that arc, is one of the most crucial processes of the entire scheme. By establishing a correspondence between the corners of the frame – in pixel positions, on the captured image of the feature, and the *known* coordinates those same corners have – by definition – when the feature is plotted, the code reader is able to perform the following tasks: (a) transform the image in order to remove the distortions introduced by perspective projection, i.e. *warping*, or the more descriptive term *perspective correction*, and (b) use the above correspondence, in the form of a variable transformation, to go back and forth between *pixel* and *plot* coordinates during the tracing of the pattern.

1.4.2 Operation Principle

The tracing of the pattern and the retrieval of the projected trajectory relies on a mechanism known as *chaotic synchronisation*. When two nonlinear, dissipative, self-sustained oscillators are coupled via a shared component, their rhythms adjust. *Rhythm adjustment*, in the present context, means that after a certain period of time, their non-interacting components evolve *synchronously*, i.e. they coincide. The type of synchronisation the NLCode utilises, involves the *unidirectional* coupling of two systems, such that one of them affects the other, but is not itself affected by it.

Specifically, one of the components of the 2D projected trajectory plotted, acts as the shared component of two systems: The first one is that which created the trajectory in the first place, called *drive*. The second system is the one implemented for the retrieval of the trajectory, called *response*. The response system is given random initial conditions within a reasonable range of values, and is integrated while accessing the image of the NLCode, in search of the pattern. The response trajectory is drawn into the attractor of the system, and that forces it to cross paths with the pattern. When that happens, the response system is driven by one of the components of the printed trajectory and starts synchronising with it. The synchronisation allows the response system to trace the pattern and retrieve the sequence of points in it.

2

Theoretical Background: Nonlinear Dynamics & Chaos

A *dynamical system* is the mathematical description of the evolution of a physical system's state in time. Time is considered an independent variable which can be continuous or discrete. In the former case the dynamical system is referred to as *Continuous-time (CT)*, and in the latter case as *Discrete-time (DT)*. The states of the physical system are represented by points in an m -dimensional space called *state space*. The dependent variable of the system – an m -dimensional vector called *state variable* – is the symbolic representation of those points in state space. The evolution of the state variable in time is governed by a *deterministic rule*, also known as the *time-evolution law* of the system, based on the known state(s) at one, several, or all previous times. If time can take both negative and nonnegative values, i.e. extend into the past as well as into the future, then the dynamical system is said to be *invertible*, whereas if time is restricted to taking only nonnegative values the system is *noninvertible* (Katok & Hasselblatt, 1995). Lastly, if the time-evolution law is *not* explicitly dependent on time the system is called *autonomous*, and if there is an explicit dependence on time, *nonautonomous*.

The dynamical systems this study is mainly focused on are CT, autonomous and invertible systems in a *three-dimensions (3D)* state space, whose deterministic rule is described by systems of 1st order *Ordinary Differential Equations (ODEs)* written in vector form as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{p}_S; \mathbf{x}) , \quad (2.1)$$

where $\mathbf{x}(t) = (x^1(t), x^2(t), x^3(t))^T$ is the system's 3D state vector, $\dot{\mathbf{x}}(t) \in \mathbb{R}^3$ is \mathbf{x} 's first order derivative with respect to time $t \in \mathbb{R}$, and $\mathbf{f} = (f^1, f^2, f^3)^T$ a continuously differentiable vector field – a C^1 function $\mathbf{f} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, with a C^1 inverse for all t . Vector $\mathbf{p}_S = (p_S^1, \dots, p_S^k)$ represents the set of real valued parameters influencing the system's behaviour, which for this reason are often referred to as *system*,

control or bifurcation parameters⁵. In its regular, expanded form, Sys. 2.1 becomes

$$\begin{aligned}\dot{x}^1 &= f^1(x^1, x^2, x^3) \\ \dot{x}^2 &= f^2(x^1, x^2, x^3) \\ \dot{x}^3 &= f^3(x^1, x^2, x^3),\end{aligned}\tag{2.2}$$

where the *parameter vector* \mathbf{p}_S has been omitted, as is usually done, for clarity.

2.1 Solution of a System of ODEs

For a given set of *Initial Conditions (ICs)* $\mathbf{x}_0 = (x_0^1, x_0^2, x_0^3)^\top$, where $x_0^j = x^j(t_0)$, $j = 1, \dots, m (= 3)$, and t_0 some initial point in time, the solution of a system of ODEs is denoted by either $\mathbf{x}(\mathbf{x}_0; t)$ or $\phi(\mathbf{x}_0; t)$ when it is thought of as a function of time parametrised by \mathbf{x}_0 that satisfies Sys. 2.2, and by $\phi_t(\mathbf{x}_0)$ when it is considered a function of \mathbf{x}_0 parametrised by time t , that is, $\phi_t : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ (see e.g. Cvitanovic et al., 2020)⁶. The latter two types of notation are often used to denote *flows*, and $\mathbf{x}(\mathbf{x}_0; t)$ is one, since it possesses the three properties characterising flows, namely (i) $\mathbf{x}(\mathbf{x}_0; t)$ is C^k for $k \geq 1$, (ii) $\mathbf{x}(\mathbf{x}_0; t_0) = \mathbf{x}_0$, and (iii) $\mathbf{x}(\mathbf{x}_0; t + \tau) = \mathbf{x}(\mathbf{x}(\mathbf{x}_0; \tau); t)$ ⁷ (Cvitanovic et al., 2020; Wiggins, 2003).

Depending on the type of vector field $\mathbf{f}(\mathbf{p}_S; \mathbf{x})$ is, the solution of a system of ODEs such as Sys. 2.2 can generally be obtained in integral form by any of a large number of *analytical* methods/techniques available, or, as is the case for most high-order, nonlinear systems of ODEs, in an approximate form by various types of *numerical* methods that have been developed for this purpose.

One such numerical method, and certainly one of the most widely used methods in Science and Engineering applications, takes its name from the German mathematicians Carl Runge (1856 – 1927) and Martin Wilhelm Kutta (1867 – 1944) who developed it around 1900 (O'Connor & Robertson, n.d.). The *4th-Order Runge-Kutta (RK4)* method, as its name suggests, numerically integrates the dependent

⁵According to Sprott (2003) the term *bifurcation* is of Latin origin and means *two branches*. Bifurcation theory studies the qualitative/structural changes effected to a system as the parameters entering its mathematical description vary (Devaney, 1989; Sprott, 2003).

⁶One of the reasons for the introduction of the ϕ and ϕ_t symbols is that when \mathbf{x}_0 is treated as a variable, it might be more convenient to denote as \mathbf{x} instead, in which case function and argument would be assigned the same symbol leading to the ambiguous notation $\mathbf{x}(\mathbf{x}; t)$ (see Wiggins, 2003, p. 93) also see footnote 7).

⁷Another reason to use ϕ_t to represent flows, is to avoid cumbersome expressions such as $\mathbf{x}(\mathbf{x}(\mathbf{x}_0; \tau); t)$ to denote functional composition. Using ϕ_t and denoting functional composition by $\cdot \circ \cdot$, the latter expression becomes $\phi_{t+\tau}(\mathbf{x}_0) = \phi_t \circ \phi_\tau(\mathbf{x}_0)$ (see e.g. Cvitanovic et al., 2020).

variables of a system of ODEs, taking into account only up to fourth-order terms in the iteration step (increment) of the independent variable. Since the present study mostly treats 3D nonlinear dynamical systems that cannot be solved analytically, their solutions are obtained numerically, using the RK4 scheme.

The *recurrence relation* that defines the RK4 method in the general case of a *nonautonomous* system of ODEs of the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ in m dimensions, reads, in vector form,

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathcal{R}\mathcal{K}, \quad (2.3)$$

where \mathbf{x}_{n+1} is each new value of the state vector calculated based on the previous one \mathbf{x}_n , \mathcal{R} is an $m \times 4$ matrix whose i -th column's entries are the components of the m -dimensional column vector $\mathbf{r}_i = (r_i^1, \dots, r_i^m)^\top$, and \mathcal{K} is the constant column vector $\frac{1}{6}(1, 2, 2, 1)^\top$. The vectors \mathbf{r}_i that fill up matrix \mathcal{R} for $i = 1, 2, 3, 4$ are defined as follows:

$$\begin{aligned} \mathbf{r}_1 &= (r_1^1, \dots, r_1^m)^\top = \Delta t \mathbf{f}(\mathbf{x}_n, t_n) \\ \mathbf{r}_2 &= (r_2^1, \dots, r_2^m)^\top = \Delta t \mathbf{f}\left(\mathbf{x}_n + \frac{1}{2}\mathbf{r}_1, t_n + \frac{1}{2}\Delta t\right) \\ \mathbf{r}_3 &= (r_3^1, \dots, r_3^m)^\top = \Delta t \mathbf{f}\left(\mathbf{x}_n + \frac{1}{2}\mathbf{r}_2, t_n + \frac{1}{2}\Delta t\right) \\ \mathbf{r}_4 &= (r_4^1, \dots, r_4^m)^\top = \Delta t \mathbf{f}(\mathbf{x}_n + \mathbf{r}_3, t_n + \Delta t), \end{aligned} \quad (2.4)$$

where Δt is the iteration step in time previously mentioned. Note that the second argument of vector field \mathbf{f} in Eqs. 2.4, which represents time t , is only relevant when the treated system is nonautonomous. This dependence of \mathbf{f} on time is included here for completeness and will be absent in the rest of this presentation which treats only autonomous systems. Using Eq. 2.3 and Eqs. 2.4, the j -th component of \mathbf{x}_{n+1} can now be written as

$$x_{n+1}^j = x_n^j + \frac{1}{6} \left(r_1^j + 2r_2^j + 2r_3^j + r_4^j \right), \quad j = 1, \dots, m. \quad (2.5)$$

A numerical solution of a 3D ($m = 3$) system like Sys. 2.2 for a specific \mathbf{f} and known parameter vector \mathbf{p}_5 is obtained using the RK4 scheme as follows:

1. For $n = 0$, which corresponds to some initial value of time t_0 (usually $t_0 = 0$), the initial state vector $\mathbf{x}_0 = (x_0^1, x_0^2, x_0^3)$ and the *constant* iteration step $\Delta t = t_{n+1} - t_n$, $n = 0, 1, \dots$, are given values appropriate for the system and the particular application.

2. Using the values specified in the previous step, vectors r_i for $i = 1, 2, 3, 4$ are calculated from Eqs. 2.4.
3. Using the values determined for the components r_i^j , $i = 1, 2, 3, 4$, $j = 1, 2, 3$ of the r -vectors and the ICs, the state vector is updated to $x_1 = (x_1^1, x_1^2, x_1^3)$ using Eq. 2.3 or Eqs. 2.5.
4. The process is repeated for $n = 1, \dots, N$, where N is the number of iterations required for time t to reach a predefined value $t_N = N\Delta t$.

The output of an algorithm implementing the RK4 scheme is a sequence $\{x_n\}$ of $N + 1$ points (x_n^1, x_n^2, x_n^3) in the system's 3D state space – including the ICs, each corresponding to a time instant $t_n = n\Delta t$, $n = 1, \dots, N$. The solution $x(x_0; t)$ of Sys. 2.2 is called *trajectory* or *state curve* of the system passing through x_0 at t_0 , and the sequence of pairs (x_n, t_n) – particularly their graph, when plotting x_n over t_n is possible – is called an *integral curve* of the system, initiating at (x_0, t_0) . Finally, sequence $\{x_n\}$ is sometimes referred to as the system's *orbit* passing through (x_0^1, x_0^2, x_0^3) (Wiggins, 2003). It is worth noting that while the above terminology is valuable in situations where the distinction between, say a trajectory and an orbit is important, it is not at all uncommon for some of the terms just introduced to be used almost interchangeably.

Figure 2.1 illustrates what has been discussed in this section with a plot of the 3D solution of the famous Lorenz system, for the parameter values known to produce the attractor known as *Lorenz attractor*⁸.

Note that the trajectories of an autonomous deterministic system cannot (self-)cross in state space, except at an unstable equilibrium point (Sec. A.4). A heuristic argument for this statement, which is formally substantiated by *Existence and Uniqueness Theorems* (see e.g. Hirsch et al., 2004, Sec. 7.2), comes from considering what a potential intersection of two trajectories, or a trajectory with itself, would imply. In both cases, a trajectory arriving at such point of intersection, could take one of two possible directions. If the choice of which one to take was a probabilistic one, then the system would not be deterministic; and if it was dependent on the time of intersection, then the system would not be autonomous. As will become obvious in Chapter 4, one of the major obstacles faced in this study, is that this “no-intersection” rule collapses in 2D projections of the trajectories (see e.g. Fig. 3.5).

⁸ The concept of an *attractor* is introduced in Secs. 2.5 and 2.6, and the Lorenz system and its attractor are discussed in detail in Sec. 3.3.

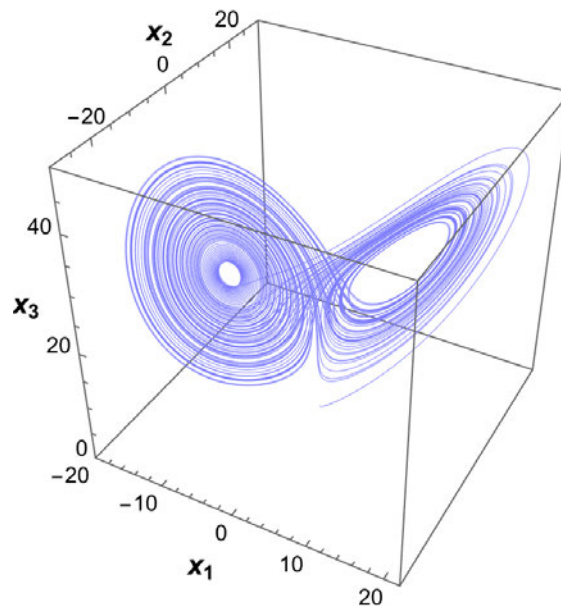


Figure 2.1: Lorenz attractor: Solution of the Lorenz system (Sys. 3.45) using the RK4 scheme presented in this section. The parameter values are the ones originally used by E. Lorenz (1963) (see Sec. 3.3, Eq. 3.46a), with ICs $x_0 = (10.0, 10.0, 10.0)$, iteration step $\Delta t = 10^{-3}$ seconds, and a number of iterations $N = 100,000$.

Solving a system of ODEs like Sys. 2.2 for a given set of ICs is only a first step toward an understanding of the dynamics that govern the system's behaviour which can, and quite often does, become very complex. Generally speaking, complexity in dynamical systems, in the form known as *chaos*, arises from the nonlinearities in the deterministic law of a system, i.e. in the case of Sys. 2.2, the nonlinear terms in x present in vector field $f(p_S; x)$. Apart from nonlinearity, chaos additionally requires – among other conditions that will be discussed in some detail in Sec. 2.6 – that a CT, *autonomous* system of ODEs be *at least three-dimensional*⁹. The dynamical systems treated from this point onward will be considered nonlinear, unless otherwise specified.

⁹ The Poincaré-Bendixson theorem, named after French mathematician and physicist Jules Henri Poincaré (1854 – 1912) and Swedish mathematician Ivar Otto Bendixson (1861 – 1935), restricts the types of asymptotic (or limiting) solutions of 2D systems to fixed points and closed trajectories. By doing that, it rules out the possibility of chaotic behaviour in 2D, therefore setting the implicit requirement of at least three dimensions for chaos to appear (Hirsch et al., 2004; Sprott, 2010).

2.2 Volume Deformation in State Space

Consider [Sys. 2.1](#) defined on \mathbb{R}^3 and a 3D *infinitesimal neighbourhood* U_0 of some state vector x_0 at time t_0 ([Fig. 2.2](#)). Let δx_0 be *any infinitesimal displacement* at x_0 , such that $x_0 + \delta x_0$ is always in U_0 ; U_0 then contains any and all points $x_0 + \delta x_0$ that are infinitesimally close to x_0 at time t_0 . Assuming that x_0 and $x_0 + \delta x_0$ are the ICs of two of [Sys. 2.1](#)'s distinct solutions, then as illustrated in [Fig. 2.2](#), at time t those two points will have transformed by the system's flow to points $x(x_0; t)$ and $x(x_0 + \delta x_0; t)$ respectively, and $\delta x_0 = \delta x(x_0; t_0)$ to $\delta x(x_0; t)$, such that

$$\delta x(x_0; t) = x(x_0 + \delta x_0; t) - x(x_0; t). \quad (2.6)$$

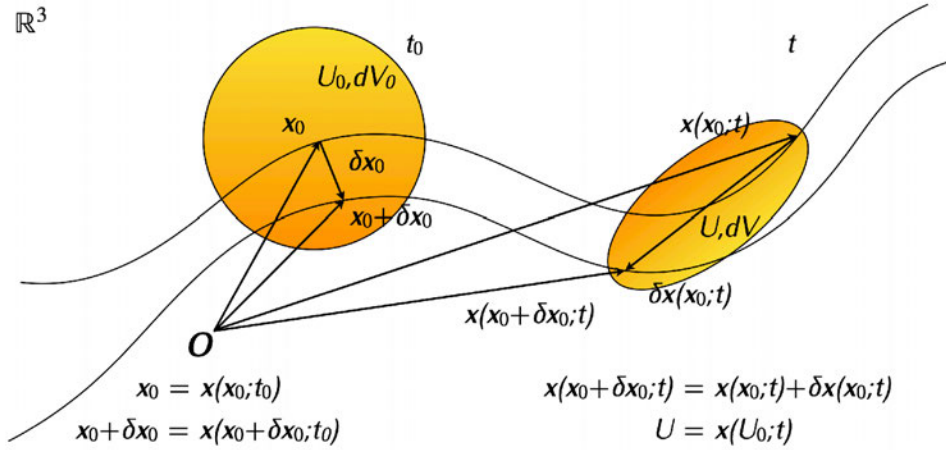


Figure 2.2: Deformation of an infinitesimal neighbourhood in [Sys. 2.1](#)'s state space: An infinitesimal neighbourhood U_0 of initial volume dV_0 at time t_0 , is deformed under the system's flow ϕ_t to an ellipsoid U of volume dV (also see [Fig. 2.3](#)) at a finite – but not “too long” – time t . Within U_0 , two ICs x_0 and $x_0 + \delta x_0$ separated by an infinitesimal displacement δx_0 are transformed by $\phi_t(x_0)$, and as $\delta x_0 = \delta x(x_0; t_0)$ becomes $\delta x(x_0; t)$ at time t , they become $x(x_0; t)$ and $x(x_0 + \delta x_0; t) = x(x_0; t) + \delta x(x_0; t)$ respectively.

From [Eq. 2.6](#), it is possible to devise a system of ODEs for the displacement $\delta x(x_0; t)$ of $x(x_0; t)$'s infinitesimally close solution $x(x_0 + \delta x_0; t)$. That system has the form

$$\dot{\delta x}(x_0; t) = Df(x)|_{x=x(x_0; t)} \delta x(x_0; t) + \mathcal{O}(|\delta x(x_0; t)|^2), \quad (2.7)$$

and is obtained by differentiating [Sys. 2.6](#) with respect to time t , replacing $\dot{x}(x_0 + \delta x_0; t)$ with $f(x(x_0 + \delta x_0; t))$ by virtue of $x(x_0 + \delta x_0; t)$ being a solution of

Sys. 2.1 – therefore satisfying it, taking $\mathbf{f}(\mathbf{x}(x_0 + \delta x_0; t))$'s first-order Taylor expansion near $\mathbf{x}(x_0; t)$ – but keeping the terms $\mathcal{O}(|\delta x(x_0; t)|^2)$ of second and higher order in δx , and after a final rearrangement of terms which, at this point, all refer to the trajectory that passes from x_0 at t_0 .

Following the terminology and notation used in Cvitanovic et al. (2020), $D\mathbf{f}(\mathbf{x})$ in **Sys. 2.7** is an $m \times m$ (3×3) matrix also denoted by $\mathcal{A}(\mathbf{x})$, defined as

$$\mathcal{A}(\mathbf{x}) = D\mathbf{f}(\mathbf{x}) = \frac{\partial(f^1, \dots, f^m)}{\partial(x^1, \dots, x^m)} \quad (2.8)$$

which, in **Sys. 2.7**, is calculated along the trajectory $\mathbf{x}(x_0; t)$. Despite $\mathcal{A}(\mathbf{x})$ having the form of what is known to most of us as the *Jacobian matrix*¹⁰ of a vector field – \mathbf{f} in this case, the authors in Cvitanovic et al. (2020) vehemently argue that this is not the case, and follow Tabor (1989) in referring to it as the *stability matrix*. For reasons that will become much clearer in **Sec. 2.3**, $\mathcal{A}(\mathbf{x})$ will be referred to as the stability matrix here as well, in what is hoped to be a successful attempt to clearly distinguish it from the *fundamental solution matrix* introduced below (after Wiggins, 2003) which, according to Cvitanovic et al. (2020), should be called the Jacobian matrix of the flow.

By performing a different sequence of operations on **Eq. 2.6**, it is also possible to obtain a relation that expresses $\delta x(x_0; t)$ in terms of $\mathbf{x}(x_0 + \delta x_0; t)$'s initial displacement from $\mathbf{x}(x_0; t)$, δx_0 . This relation is of the form

$$\delta x(x_0; t) = D\phi_t(x_0) \delta x_0 + \mathcal{O}(|\delta x_0|^2), \quad (2.9)$$

and is obtained by first substituting $\mathbf{x}(x_0 + \delta x_0; t)$ on the right hand side of **Eq. 2.6** by its notational equivalent $\phi_t(x_0 + \delta x_0)$, then taking the same term's first-order Taylor expansion around x_0 – but keeping the terms $\mathcal{O}(|\delta x_0|^2)$ of second and higher order in δx_0 , and noticing that the Taylor series' first term $\phi_t(x_0)$ and **Eq. 2.6**'s original term $\mathbf{x}(x_0; t)$ cancel each other out.

Equation 2.9 introduces the $m \times m$ (3×3) matrix $D\phi(x_0)$ known as the *fundamental solution matrix* of **Sys. 2.1**, also denoted by $\mathcal{X}_t(x_0)$ and defined as

$$\mathcal{X}_t(x_0) = D\phi_t(x_0) = \frac{\partial(\phi_t^1, \dots, \phi_t^m)}{\partial(x_0^1, \dots, x_0^m)}. \quad (2.10)$$

¹⁰According to J.C. Sprott (2003, p. 74), the term *Jacobian matrix* “was coined by the eccentric and gifted English mathematician James Joseph Sylvester (1814 – 1897) in 1852 in honour of the German mathematician Carl Gustav Jacob Jacobi (1804-1851)” (for the biographies of the latter two mathematicians the reader is referred to O'Connor & Robertson, n.d.).

It should be noted that [Sys. 2.7](#) and [Eq. 2.9](#) are most commonly used in situations where the problem statement permits the omission of their nonlinear terms. It is therefore in the context of problems of this kind that matrices $\mathcal{A}(\mathbf{x})$ and $\mathcal{X}_t(\mathbf{x}_0)$ are referred to as *stability* and *fundamental solution* matrix respectively. Using the *linearized* versions of [Sys. 2.7](#) and [Eq. 2.9](#) in such a context, it can be shown (see e.g. Cvitanovic et al., 2020) that $\mathcal{A}(\mathbf{x})$ and $\mathcal{X}_t(\mathbf{x}_0)$ are related by a system of ODEs which, in matrix form reads

$$\dot{\mathcal{X}}_t(\mathbf{x}_0) = \mathcal{A}(\mathbf{x}) \mathcal{X}_t(\mathbf{x}_0). \quad (2.11)$$

In the general case where the solution of [Sys. 2.1](#) is not known analytically, which means that [Eq. 2.10](#) is of no practical use, [Sys. 2.11](#) with $\mathcal{X}_{t_0}(\mathbf{x}_0) = \mathbb{I}$ as IC, where \mathbb{I} is the identity matrix, is solved numerically alongside [Sys. 2.1](#) (Cvitanovic et al., 2020). A special case where the analytical solution of [Sys. 2.11](#) is straightforward is only touched upon in a later subsection discussing [Sys. 2.1](#)'s equilibrium solutions (see discussion following [Eq. 2.18](#), [Sec. 2.3](#)).

A rather intuitive physical interpretation of the two matrices $\mathcal{A}(\mathbf{x})$ and $\mathcal{X}_t(\mathbf{x}_0)$, comes from the linearised versions of [Sys. 2.7](#) and [Eq. 2.9](#) respectively. The two matrices describe the *deformation of the infinitesimal neighbourhood U along trajectory $\mathbf{x}(\mathbf{x}_0; t)$, under the flow $\phi_t(\mathbf{x}_0)$ generated by vector field $\mathbf{f}(\mathbf{x})$* , from two different perspectives¹¹: The stability matrix $\mathcal{A}(\mathbf{x})$ describes, as the linearised version of [Sys. 2.7](#) suggests, the *instantaneous rate of deformation of U* , and by virtue of the linear version of [Eq. 2.9](#), the fundamental solution matrix $\mathcal{X}_t(\mathbf{x}_0)$ describes the *finite-time deformation of U* .

Dissipation

Dynamical systems are categorised based on whether the volume of an infinitesimal neighbourhood in state space is increased, preserved, or decreased by the deformation of state space under the system's flow. In the first case the systems are called *volume expanding*, in the second *volume preserving* or most commonly *conservative*, and in the third case which is of particular interest here, the systems are called *volume contracting* or *dissipative*.

In order to determine which of the above categories a system falls into, one is required to calculate the *rate of volume change* dV/dt – its sign, to be precise –

¹¹[System 2.7](#) and [Eq. 2.9](#) describe the dynamics of a single flow from two different perspectives which are known, especially in Continuum/Fluid Mechanics, as *Eulerian* and *Lagrangian* points of view respectively (see e.g. Cvitanovic et al., 2020).

either at a single point in state space, in which case one refers to a *local property*, or for all points in state space if that is possible, thus characterising a *global property* of the system (Cvitanovic et al., 2020). S. Wiggins offers a quite intuitive derivation of a formula for dV/dt (Wiggins, 2003, Sec. 7.6), however, one can reach the same goal in one step by making use of the *Reynolds Transport Theorem*. For any quantity $B = B(\mathbf{x}, t)$ – not strictly a scalar – associated with a flow with a velocity field $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$, which is integrated over a time-dependent volume element $V(t)$, the Reynolds transport theorem is expressed as follows:

$$\frac{d}{dt} \left(\iiint_{V(t)} B(\mathbf{x}, t) dV \right) = \iiint_{V(t)} \left(\frac{\partial B(\mathbf{x}, t)}{\partial t} + \nabla \cdot [B(\mathbf{x}, t) \mathbf{u}(\mathbf{x}, t)] \right) dV, \quad (2.12)$$

where ∇ is the *nabla operator*. This theorem is a 3D generalisation of the Leibnitz rule in 1D; it describes the rule according to which a total derivative operator with respect to some variable – here d/dt – can be moved inside an integral “when both the integrand and the limits of integration depend on that variable” (Leal, 2007, p. 22). For the present purposes $B(\mathbf{x}, t)$ is set to 1 so that the integral on the left hand side of Eq. 2.12 becomes the volume element $V(t)$ itself, and the velocity field $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is the system’s (autonomous) vector field $\mathbf{f}(\mathbf{x})$, such that the final formula for the rate of volume change dV/dt is

$$\frac{dV(t)}{dt} = \iiint_{V(t)} \nabla \cdot \mathbf{f}(\mathbf{x}) dV, \quad (2.13)$$

where

$$\nabla \cdot \mathbf{f} = \sum_{j=1}^m \frac{\partial f_j}{\partial x_j} = \text{Tr}(\mathcal{A}) \quad (2.14)$$

is the divergence of $\mathbf{f}(\mathbf{x})$, which is equal to the *trace* of the stability matrix $\mathcal{A}(\mathbf{x})$ (Eq. 2.8) denoted by $\text{Tr}(\cdot)$.

From Eq. 2.13 one concludes that depending on whether $\nabla \cdot \mathbf{f}$ is positive, equal to zero or negative in some region of state space, the dynamical system under consideration is volume expanding, conservative or dissipative in that region respectively (Ott, 1993).¹²

¹² It was pointed out as early as 1992 (X. Gan et al., 2021, and references therein), and was certainly encountered by D. Li (2008), a case which will be discussed in the Appendix, that the *divergence criterion* might not be conclusive in every possible scenario, in which case the dissipation of a system can only be determined via alternative routes. A recent treatment of this issue for 2D systems can be found in X. Gan et al. (2021).

2.3 Equilibrium Solutions & Local Stability

The simplest type of solutions possessed by *autonomous* dynamical systems are those that do not change in time, or equivalently, solutions for which the system's vector field vanishes, i.e.

$$\left. \frac{dx}{dt} \right|_{x=x^*} = \mathbf{f}(x^*) = \mathbf{0}, \quad (2.15)$$

where

$$x^* = x(x^*; t) = \phi_t(x^*), \text{ for all } t, \quad (2.16)$$

denotes such a solution. This type of solutions are known as *equilibrium* or *stationary solutions* and *fixed points*, among quite a few other terms used in the literature (see e.g. Perko, 2001; Wiggins, 2003).

All solutions of dynamical systems are characterised by a property known as *stability*. In qualitative terms, a solution $x(x_0; t)$ is said to be (*asymptotically*) *stable* if all other solutions that start arbitrarily close to $x(x_0; t)$, (converge) remain arbitrarily close to it (as $t \rightarrow \infty$) for all $t > t_0$. A solution that is not stable is referred to as an *unstable solution* (Wiggins, 2003).

Albeit not particularly technical in their description, the above definitions suggest that the stability of a solution of an autonomous dynamical system like [Sys. 2.1](#) can, in principle, be inferred from the solution of [Sys. 2.7](#), provided that can be obtained, of course. Unfortunately, [Sys. 2.7](#) is not linear – due to the nonlinear terms in $\delta x(x_0; t)$, and even if it were, it does not have constant coefficients – since matrix $\mathcal{A}(x)$ ([Eq. 2.8](#)), through $x = x(x_0; t)$, depends not only on x_0 but on t as well. The first obstacle is overcome in matters regarding stability, because the requirement that solutions remain arbitrarily close, i.e. $\delta x(x_0; t)$ arbitrarily small, allow the nonlinear terms $\mathcal{O}(|\delta x(x_0; t)|^2)$ to be considered negligible and omitted as such. In fact, stability problems are the most characteristic examples of the type of problems mentioned earlier, that make use of the linearised versions of [Sys. 2.7](#) and [Eq. 2.9](#). The second hurdle, i.e. that of $\mathcal{A}(x)$'s time-dependence, which still implies that a solution to [Sys. 2.7](#) is not guaranteed, is avoided altogether when the system refers to the displacement $\delta x(x^*; t)$ between [Sys. 2.1](#)'s equilibrium solution $x^* = x(x^*; t)$ ([Eq. 2.16](#)) and its close neighbour $x(x^* + \delta x^*; t)$.

Under the above conditions, and using [Eq. 2.8](#) to replace $D\mathbf{f}(x)|_{x=x^*}$ with $\mathcal{A}(x^*)$, [Sys. 2.7](#) becomes

$$\dot{\delta x}(x^*; t) = \mathcal{A}(x^*) \delta x(x^*; t), \quad \delta x(x^*; t) \text{ arbitrarily small for all } t, \quad (2.17)$$

which is a linear system of ODEs with constant coefficients formally known as *system of variations*, or when separately referring to its components, as *variational or stability equations* (Cvitanovic et al., 2020), although it is more often referred to as *associated linear system*. Provided $\mathcal{A}(x^*)$ is *nonsingular*, i.e. $\det \mathcal{A}(x^*) \neq 0$, Sys. 2.17 has the unique equilibrium solution $\delta x^* = \delta x(x^*; t) = \mathbf{0}$, while its general solution in terms of the matrix exponential is

$$\delta x(x^*; t) = e^{\mathcal{A}(x^*)t} \delta x_0, \quad \delta x(x^*; t) \text{ arbitrarily small for all } t, \quad (2.18)$$

where $\delta x_0 = \delta x(x^*; t_0)$. Note that when comparing Eq. 2.18 with the linearised version of Eq. 2.9 around $x^* = x(x^*; t)$, which is $\delta x(x^*; t) = \mathcal{X}_t(x^*) \delta x_0$, where $\mathcal{X}_t(x^*)$ has replaced $D\phi(x_0)$ using Eq. 2.10, it is seen that *in this particular case*, the fundamental solution matrix is given by $\mathcal{X}_t(x^*) = e^{\mathcal{A}(x^*)t}$. The same result is obtained by also noticing that Sys. 2.11's general solution around a fixed point is $\mathcal{X}_t(x^*) = e^{\mathcal{A}(x^*)t} \mathcal{X}_{t_0}(x^*)$ or, since $\mathcal{X}_{t_0}(x^*) = \mathbb{I}$, $\mathcal{X}_t(x^*) = e^{\mathcal{A}(x^*)t}$.

Due to the condition that $\delta x(x^*; t)$ be arbitrarily small, Eq. 2.18 describes the behaviour of the linear Sys. 2.17's solutions in the neighbourhood of its fixed point $\delta x^* = \mathbf{0}$. However, as already implied earlier, it also describes the behaviour of the solutions of the nonlinear Sys. 2.1 in the neighbourhood of its equilibrium point $x^* = x(x^*; t)$, at least in qualitative terms. To make things more precise, one can invoke two important theorems of Dynamics that prove, in conjunction, that the stability of a fixed point x^* of a nonlinear vector field $f(x)$ (Sys. 2.1) is the same as that of the linear vector field $Df(x^*)$'s fixed point at the origin (Sys. 2.17), *provided none of the eigenvalues of the linear field have zero real parts*.

The first theorem is known as the *Stable Manifold Theorem* and says that the stable and unstable sets of the nonlinear vector field's equilibrium point are smooth manifolds whose tangent spaces are the stable and unstable subspaces respectively, of the linear vector field's fixed point (Perko, 2001). The second theorem is named after mathematicians Philip Hartman and David M. Grobman who proved it independently in Hartman (1963) and (Grobman, 1959) respectively. The *Hartman-Grobman Theorem* says that the flow of the nonlinear vector field in the neighbourhood of its fixed point is qualitatively the same as the flow of the linear field near its own fixed point at the origin (Perko, 2001; Wiggins, 2003).

A first step toward the characterisation of a nonlinear system's equilibrium solution in terms of stability, is to note that the two theorems cited above have already created a category of fixed points based on the eigenvalues of matrix

$Df(\mathbf{x}^*) = \mathcal{A}(\mathbf{x}^*)$: The equilibrium solutions at which none of the eigenvalues of $\mathcal{A}(\mathbf{x}^*)$ have zero real parts are called *hyperbolic equilibrium points*. If even one of $\mathcal{A}(\mathbf{x}^*)$'s eigenvalues has zero real part, \mathbf{x}^* is *nonhyperbolic* (Perko, 2001).

The fact alone that the two cited theorems apply only to hyperbolic points suggests that the process of identifying the stability type of \mathbf{x}^* will probably be more involved when that fixed point is nonhyperbolic. This is indeed the case; *Lyapunov's Direct Method*, as it is often referred to (in Salle & Lefschetz, 1961, for example), is one of the basic methods used for determining the stability of nonhyperbolic points (also see Perko, 2001). Its treatment, however, is far beyond the scope of this thesis, which in turn means that any mention of equilibrium solutions from this point onward will strictly refer to *hyperbolic equilibrium points*, unless otherwise specified.

As it turns out, the eigenvalues of matrix $\mathcal{A}(\mathbf{x}^*)$ also enable a finer categorisation of hyperbolic equilibrium points into different stability types. In fact, this is the reason $\mathcal{A}(\mathbf{x}^*)$ is called the stability matrix. It is therefore worth recalling a few simple facts about eigenvalues from Linear Algebra: The eigenvalues λ_j^* of the 3×3 matrix $\mathcal{A}(\mathbf{x}^*)$ are the roots of $\mathcal{A}(\mathbf{x}^*)$'s *characteristic polynomial* $\det[\mathcal{A}(\mathbf{x}^*) - \lambda \mathbb{I}]$, i.e. the solutions of equation

$$\det[\mathcal{A}(\mathbf{x}^*) - \lambda \mathbb{I}] = 0, \quad (2.19)$$

where $\det[\cdot]$ denotes the determinant of a matrix. The degree of $\mathcal{A}(\mathbf{x}^*)$'s characteristic polynomial is 3, i.e. the same as the order of the matrix, which means that $\mathcal{A}(\mathbf{x}^*)$ has *three eigenvalues*. Moreover, since $\mathcal{A}(\mathbf{x}^*)$ is a real matrix, i.e. its characteristic polynomial has real coefficients, *any complex eigenvalues will occur in complex-conjugate pairs*. As a result, one of $\mathcal{A}(\mathbf{x}^*)$'s eigenvalues is definitely real, and the other two are either real, or a complex-conjugate pair.

Table 2.1 summarises the rules that determine the stability of a hyperbolic equilibrium solution \mathbf{x}^* of a nonlinear and autonomous dynamical system in 3D, based on the eigenvalues of the stability matrix $\mathcal{A}(\mathbf{x}^*)$. Letting μ_j^* and ω_j^* denote the real and imaginary parts of $\mathcal{A}(\mathbf{x}^*)$'s eigenvalues λ_j^* respectively, i.e.

$$\lambda_j^* = \mu_j^* \pm i \omega_j^*, \quad j = 1, 2, 3, \quad (2.20)$$

where i is the imaginary unit, these rules are the following¹³:

- The existence of even one eigenvalue with zero real part creates the category of *nonhyperbolic* points; if $\mu_j^* \neq 0$ for all j , the fixed point is *hyperbolic*.

¹³Further details can be found in Perko (2001, chap. 2).

- A hyperbolic fixed point x^* is called a *saddle* and it is *unstable*, when there is at least one eigenvalue with a positive real part, and one with a negative real part; the third eigenvalue must be non-zero to ensure that x^* is hyperbolic.
- A hyperbolic fixed point x^* is called a *source* and is *unstable*, when all of $\mathcal{A}(x^*)$'s eigenvalues have *positive* real parts.
- A hyperbolic fixed point x^* is called a *sink* and it is *asymptotically stable*, when all of $\mathcal{A}(x^*)$'s eigenvalues have *negative* real parts.
- Hyperbolic equilibrium solutions can either be asymptotically stable, or unstable; for a fixed point to be stable *without being asymptotically stable*, at least one eigenvalue must have a zero real part – but then the point is nonhyperbolic.
- Nonhyperbolic equilibrium solutions – which are not treated here – can be stable, asymptotically stable or unstable, but the criteria determining the type of stability are the subject of more advanced methods.

Table 2.1: Stability of equilibrium solutions in 3D, autonomous and nonlinear dynamical systems. The set of parameters whose signs form the criteria that determine the stability of a fixed point x^* , are the real parts μ_j^* of the eigenvalues λ_j^* of the stability matrix $\mathcal{A}(x^*)$ defined by Eq. 2.8. The first column presents the criterion that determines whether an equilibrium solution of a system like Sys. 2.1 is hyperbolic or nonhyperbolic. Hyperbolic fixed points are classified into two finer stability types: Unstable saddles and sources and asymptotically stable sinks. The criterion determining this stability type of a hyperbolic equilibrium solution is presented in the last column. For reasons explained in the main text, the finer classification of nonhyperbolic points into three stability types – stable, asymptotically stable and unstable – is not presented here.

| $\mathcal{A}(x^*)$'s eigenvalues $\lambda_j^* = \mu_j^* \pm i \omega_j^*$ | Stability Types | | $\mu_j^* = \text{Re}(\lambda_j^*)$ |
|---|-----------------|-----------------------|--|
| $\mu_j^* \neq 0, \forall j$ | Hyperbolic | Unstable | Saddle $\mu_1^* > 0$ $\mu_2^* < 0$ $(\mu_3^* \neq 0)$ |
| | | | Source $\mu_j^* > 0, \forall j$ |
| | | Asymptotically Stable | Sink $\mu_j^* < 0, \forall j$ |
| $\mu_j^* = 0$ for at least one value of $j = 1, 2, 3$ | Nonhyperbolic | Stable | |
| | | Asymptotically Stable | |
| | | Unstable | |

2.4 Lyapunov Spectrum

One of the defining characteristics of chaos is known as *sensitive dependence on initial conditions*; any two solutions of a system exhibiting chaos that start arbitrarily close to one another, e.g. at the state vectors \mathbf{x}_0 and $\mathbf{x}_0 + \delta\mathbf{x}_0$ introduced in [Sec. 2.2](#), separate *exponentially* with time. As a consequence of that separation, small deviations from an intended initial state – an inescapable reality of practically any application of interest – are exaggerated, oftentimes dramatically, leading to *long-term unpredictability*. The quantities used to quantify this sensitive dependence on initial conditions are called *Lyapunov exponents* – often collectively referred to as the *Lyapunov spectrum* – or simply *characteristic exponents* (as in Eckmann & Ruelle, 1985). They are named after Aleksandr Mikhailovich Lyapunov (1857 – 1918), the Russian mathematician who developed the fundamental theory of stability for dynamical systems (O'Connor & Robertson, n.d.).

A first step toward a definition of the Lyapunov exponents is to express the *stretch ratio*¹⁴ of the distance between trajectories $\mathbf{x}(\mathbf{x}_0; t)$ and $\mathbf{x}(\mathbf{x}_0 + \delta\mathbf{x}_0; t)$ ([Fig. 2.2](#)) at time t to their original distance at time t_0 as

$$\frac{\|\delta\mathbf{x}(\mathbf{x}_0; t)\|}{\|\delta\mathbf{x}_0\|} = e^{L_t \cdot (t-t_0)}, \quad (2.21)$$

where $\|\cdot\|$ denotes the Euclidean vector norm, and $L_t = L_t(t_0, \mathbf{x}_0, \delta\mathbf{x}_0)$. Since the two trajectories start infinitesimally close, i.e. $\delta\mathbf{x}_0$ is arbitrarily small, [Eq. 2.9](#)'s nonlinear terms are omitted and $\delta\mathbf{x} = \mathcal{X}_t(\mathbf{x}_0) \delta\mathbf{x}_0$, which means that [Eq. 2.21](#) can be written as

$$\frac{\|\mathcal{X}_t(\mathbf{x}_0) \delta\mathbf{x}_0\|}{\|\delta\mathbf{x}_0\|} = e^{L_t \cdot (t-t_0)}. \quad (2.22)$$

The stretch ratio in the above form is sometimes also called *coefficient of expansion in the direction of $\delta\mathbf{x}_0$ along trajectory $\mathbf{x}(\mathbf{x}_0; t)$ that passes from point \mathbf{x}_0 at time t_0* (see e.g. Wiggins, 2003). Solving [Eq. 2.22](#) for L_t gives

$$L_t(t_0, \mathbf{x}_0, \delta\mathbf{x}_0) = \frac{1}{t - t_0} \ln \frac{\|\mathcal{X}_t(\mathbf{x}_0) \delta\mathbf{x}_0\|}{\|\delta\mathbf{x}_0\|}, \quad (2.23)$$

which is sometimes called *finite Lyapunov exponent*, a name that will soon be justified (see [Eq. 2.24](#)).

¹⁴In Continuum Mechanics, the ratio of the final length to the respective initial length of a deforming “material” is called *extension ratio*, *stretch ratio*, or simply *stretch*; when referring to the undeformed state, the stretch equals 1.

As will hopefully become intuitively clear in the following, the Lyapunov exponents are *as many as the dimensions of the system*, denoted here by L_j , $j = 1, \dots, m (= 3)$. The Lyapunov exponent $L_j = L(x_0, \delta x_0^j)$ in the direction of δx_0^j along trajectory $x(x_0; t)$ that passes from point x_0 at *any* time – notice in Eq. 2.24 below that t_0 is no longer a dependency – is defined as the limit of $L_t(t_0, x_0, \delta x_0^j)$ as $t \rightarrow \infty$, i.e.

$$L_j = L(x_0, \delta x_0^j) = \lim_{t \rightarrow \infty} \frac{1}{t - t_0} \ln \frac{\|\mathcal{X}_t(x_0) \delta x_0^j\|}{\|\delta x_0^j\|}, \quad j = 1, \dots, m (= 3), \quad (2.24)$$

provided, of course, that the limit exists. In fact, *Oseledec's multiplicative ergodic theorem* showed that the infinite-time limit in the above definition exists for *almost* all trajectories (Wiggins (2003), Geist et al. (1990) and references therein, most prominently Oseledets (1968) – also spelled *Oseledec*, published in Russian). Note that the superscript j in δx_0^j indicates that the Lyapunov spectrum is calculated in the directions of $m (= 3)$ vectors which, while they can quite literally be *any* vectors, can be chosen in a quite effective and intuitive way. The Lyapunov exponents are conventionally indexed in descending order with repetition (Sprott, 2010), from the least negative to the most negative, as is usually stated in order to place emphasis on their relative signs, that is,

$$L_1 \geq L_2 \geq \dots \geq L_m. \quad (2.25)$$

There is a second approach to the definition of the Lyapunov exponents, that offers a very intuitive geometric interpretation of these quantities. This approach is illustrated in Fig. 2.3 and thoroughly discussed below, following the outline provided in Geist et al. (1990). It relies on the *singular value decomposition* of the fundamental solution matrix \mathcal{X}_t , i.e. a factorisation of the form

$$\mathcal{X}_t = \mathcal{U} \mathcal{F} \mathcal{U}_0^T, \quad (2.26)$$

where for \mathcal{X}_t a real, square matrix, as is the case here, \mathcal{U}_0 and \mathcal{U} are *orthogonal matrices* whose columns, seen as vectors, form two *orthonormal bases* $\{\hat{u}_0^j\}$ and $\{\hat{u}^j\}$, $j = 1, \dots, m (= 3)$ respectively. \mathcal{F} is the *diagonal matrix* $\mathcal{F} = \text{diag}(\sigma_t^1, \sigma_t^2, \sigma_t^3)$ whose diagonal elements are called *singular values* of \mathcal{X}_t and are by definition non-negative.

Using only the orthogonality of \mathcal{U}_0 and \mathcal{U} , which means that $\mathcal{U}_0^{-1} = \mathcal{U}_0^\top$ and similarly for \mathcal{U} , and the fact that for any two matrices \mathcal{A} and \mathcal{B} the relation $(\mathcal{A}\mathcal{B})^\top = \mathcal{B}^\top \mathcal{A}^\top$ holds true, multiplying Eq. 2.26 with its transpose which is $\mathcal{X}_t^\top = (\mathcal{U}\mathcal{F}\mathcal{U}_0^\top)^\top$, from the left and from the right, leads to equations

$$\mathcal{X}_t^\top \mathcal{X}_t = \mathcal{U}_0 (\mathcal{F}^2) \mathcal{U}_0^{-1} \quad (2.27a)$$

$$\mathcal{X}_t \mathcal{X}_t^\top = \mathcal{U} (\mathcal{F}^2) \mathcal{U}^{-1}. \quad (2.27b)$$

The above equations represent a particular type of factorisation of the matrices on their left hand side which, provided all matrices involved are *square* and \mathcal{F}^2 is *diagonal* – both of which are true in this case, is called *eigendecomposition*, or *diagonalisation* of $\mathcal{X}_t^\top \mathcal{X}_t$ and $\mathcal{X}_t \mathcal{X}_t^\top$ respectively. The columns of matrices \mathcal{U}_0 and \mathcal{U} as vectors are *eigenvectors* of the respective matrices, and the diagonal elements $(\sigma_t^j)^2$ of \mathcal{F}^2 , the same matrices' corresponding eigenvalues λ_t^j , which are known to be non-negative. Note that according to the last statement, matrices $\mathcal{X}_t^\top \mathcal{X}_t$ and $\mathcal{X}_t \mathcal{X}_t^\top$ both have the same eigenvalues.

In a similar way, multiplying Eq. 2.26 from the right with \mathcal{U}_0 easily leads to equation $\mathcal{X}_t \mathcal{U}_0 = \mathcal{U}\mathcal{F}$ which can be expressed in terms of \mathcal{U}_0 and \mathcal{U} 's columns – represented by vectors $\hat{\mathbf{u}}_0^j$ and $\hat{\mathbf{u}}^j$ respectively, as

$$\mathcal{X}_t \hat{\mathbf{u}}_0^j = \sigma_t^j \hat{\mathbf{u}}^j, \quad j = 1, \dots, m(= 3). \quad (2.28)$$

The above equation shows that the action of matrix \mathcal{X}_t on vectors $\hat{\mathbf{u}}_0^j$ transforms them to vectors $\hat{\mathbf{u}}^j$, but *stretched* or *shrank* by a factor σ_t^j .

The geometric representation of the Lyapunov exponents is derived from a setting based on the above observations and illustrated in Fig. 2.3. Consider the orthonormal basis $\{\hat{\mathbf{u}}_0^j\}$ affixed at point \mathbf{x}_0 in Sys. 2.1's state space and an *infinitesimal sphere* Σ of radius $\varepsilon \ll 1$ centred at \mathbf{x}_0 at time t_0 , whose equation in vector form is $(\delta\mathbf{x}_0)^2 = \varepsilon^2$. The infinitesimal displacement $\delta\mathbf{x}_0$ was introduced in the beginning of Sec. 2.2 (Fig. 2.2), but in the present context it is best viewed as just an infinitesimal vector at \mathbf{x}_0 , confined by the equation of Σ to the surface of the sphere. If $\mathbf{s}_0^j = \varepsilon \hat{\mathbf{u}}_0^j$ are $m(= 3)$ infinitesimal vectors at \mathbf{x}_0 at time t_0 , then were it not a triviality to attribute to a geometric object of highest symmetry such as the sphere principal semi-axes, one might say that vectors \mathbf{s}_0^j were exactly that. Note that just as $\delta\mathbf{x}_0$ is transformed to $\delta\mathbf{x}$ by matrix \mathcal{X}_t (linearised version of Eq. 2.9), \mathbf{s}_0^j is transformed to some vector \mathbf{s}^j , such that $\mathbf{s}^j = \mathcal{X}_t \mathbf{s}_0^j$.

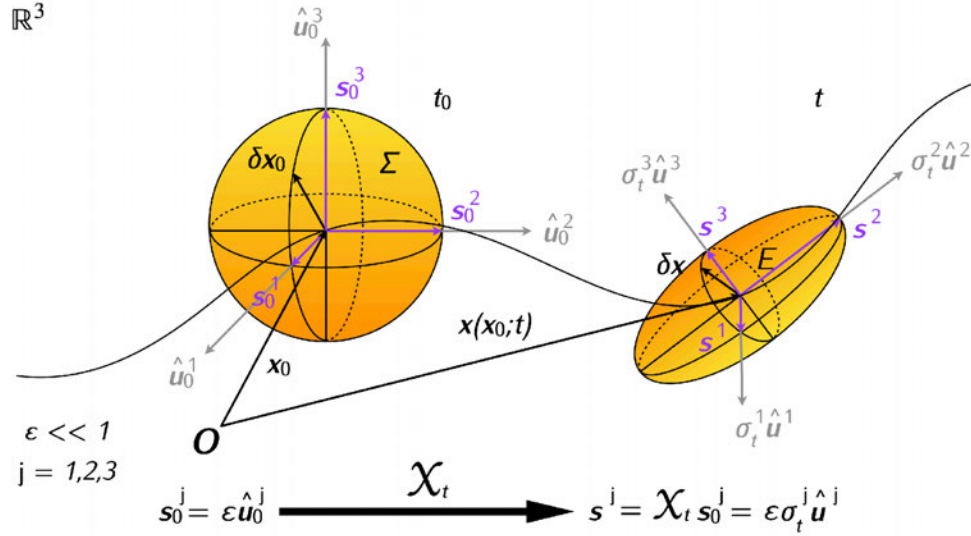


Figure 2.3: Geometric interpretation of the Lyapunov spectrum: When the Lyapunov exponents are calculated in the direction of $(\mathcal{X}_t^\top \mathcal{X}_t)$'s eigenvectors s_0^j , they express the mean logarithmic expansion (or contraction) rates of the principal semi-axes of an ellipsoid $E : (\delta x)^\top (\mathcal{X}_t \mathcal{X}_t^\top)^{-1} / \varepsilon^2 \delta x = 1$ that resulted from the deformation of an infinitesimal sphere $\Sigma : (\delta x_0)^2 = \varepsilon^2$ when acted upon by the fundamental solution matrix \mathcal{X}_t of Sys. 2.1. All of the quantities indicated in the figure have been introduced and thoroughly discussed in the main text.

Following the reasoning presented in E. N. Lorenz (1984), it is argued that at time t , the infinitesimal sphere Σ will have transformed under Sys. 2.1's flow $\phi_t(x_0)$ to an ellipsoid E centred at $x(x_0; t)$, and the following sequence of operations shows that this is indeed the case:

$$\begin{aligned}
 & \text{Sphere } \Sigma : (\delta x_0)^2 = \varepsilon^2 & (2.29) \\
 & \xrightarrow[\delta x_0 = \mathcal{X}_t^{-1} \delta x]{\text{Invertible system}} (\mathcal{X}_t^{-1} \delta x)^2 = \varepsilon^2 \\
 & \xrightarrow{\text{Dot product}} (\mathcal{X}_t^{-1} \delta x) \cdot (\mathcal{X}_t^{-1} \delta x) = \varepsilon^2 \\
 & \xrightarrow{\text{Matrix multiplication}} (\mathcal{X}_t^{-1} \delta x)^\top (\mathcal{X}_t^{-1} \delta x) = \varepsilon^2 \\
 & \xrightarrow{(AB)^\top = B^\top A^\top} (\delta x)^\top (\mathcal{X}_t^{-1})^\top \mathcal{X}_t^{-1} \delta x = \varepsilon^2 \\
 & \xrightarrow{(A^\top)^{-1} = (A^{-1})^\top} (\delta x)^\top (\mathcal{X}_t^\top)^{-1} \mathcal{X}_t^{-1} \delta x = \varepsilon^2 \\
 & \xrightarrow{(AB)^{-1} = B^{-1} A^{-1}} (\delta x)^\top (\mathcal{X}_t \mathcal{X}_t^\top)^{-1} \delta x = \varepsilon^2 \\
 & \longrightarrow (\delta x)^\top \frac{(\mathcal{X}_t \mathcal{X}_t^\top)^{-1}}{\varepsilon^2} \delta x = 1 : \text{Ellipsoid } E. & (2.30)
 \end{aligned}$$

Equation 2.30 defines an ellipsoid whose *principal semi-axes* are in the directions of the *eigenvectors* of matrix $(\mathcal{X}_t \mathcal{X}_t^\top)^{-1} / \varepsilon^2$, and have lengths that are the reciprocals of the square roots of the same matrix's *eigenvalues*. Noting that a matrix and its inverse have the same eigenvectors and reciprocal eigenvalues, and that multiplying a matrix with a scalar leaves its eigenvectors unaffected and multiplies its eigenvalues by the same scalar, leads to the following conclusions:

- Matrices $(\mathcal{X}_t \mathcal{X}_t^\top)^{-1} / \varepsilon^2$ and $(\mathcal{X}_t \mathcal{X}_t^\top)$ have the same eigenvectors which, according to the above discussion are the vectors \hat{u}^j of matrix \mathcal{U} 's columns.
- If the eigenvalues of matrix $(\mathcal{X}_t \mathcal{X}_t^\top)^{-1} / \varepsilon^2$ are denoted by λ^- , then the eigenvalues of its inverse matrix $\varepsilon^2 (\mathcal{X}_t \mathcal{X}_t^\top)$ will be $1/\lambda^-$ and $(\mathcal{X}_t \mathcal{X}_t^\top)$'s $\lambda_t^j = 1/(\varepsilon^2 \lambda^-)$.

According to the above observations, E 's principal semi-axes are $(1/\sqrt{\lambda^-}) \hat{u}^j$, or $\varepsilon \sqrt{\lambda_t^j} \hat{u}^j$, and remembering that $\lambda_t^j = (\sigma_t^j)^2$, they become $\varepsilon \sigma_t^j \hat{u}^j$. Now, multiplying both sides of Eq. 2.28 by ε gives equation $\mathcal{X}_t \varepsilon \hat{u}_0^j = \varepsilon \sigma_t^j \hat{u}^j$ whose left hand side is equal to s^j , and the right hand side is the ellipsoid's semi-axes found just above, now shown to be $s^j = \varepsilon \sigma_t^j \hat{u}^j$. In light of this, the relation $s^j = \mathcal{X}_t s_0^j$ noted earlier shows that the action of \mathcal{X}_t on the infinitesimal sphere's "*principal semi-axes*" transforms them to the principal semi-axes of ellipsoid E which apart from their orientation are also stretched (or shrank) by a factor σ_t^j .

In order to see how all of the above lead to the geometric interpretation of the Lyapunov exponents, consider L_j in the direction of the infinitesimal vectors s_0^j , along the trajectory identified by the point x_0 from which it passes (Eq. 2.24). The *squared* coefficient of expansion in this case will be

$$\frac{\|\mathcal{X}_t(x_0) s_0^j\|^2}{\|s_0^j\|^2} = (\hat{u}_0^j)^\top (\mathcal{X}_t^\top \mathcal{X}_t) \hat{u}_0^j = (\hat{u}_0^j)^\top \lambda_t^j \hat{u}_0^j = \lambda_t^j = (\sigma_t^j)^2, \quad (2.31)$$

which means that the Lyapunov exponents of Eq. 2.24 can now be written as

$$L_j = L(x_0, s_0^j) = \lim_{t \rightarrow \infty} \frac{1}{t - t_0} \ln \sigma_t^j, \quad j = 1, \dots, m (= 3). \quad (2.32)$$

In this form, i.e. when calculated in the direction of $(\mathcal{X}_t^\top \mathcal{X}_t)$'s eigenvectors s_0^j , the Lyapunov exponents express the *mean logarithmic expansion (or contraction) rates* of an ellipsoid's principal semi-axes (Geist et al., 1990), and in a more general setting, they are *global asymptotic* measures of the *average* logarithmic rates of change – growth or shrinking – of state space volumes in different directions.

Apart from quantifying the sensitivity to initial conditions and enabling the detection of chaos, the Lyapunov spectrum also relates to certain statistical properties of dynamical systems exhibiting chaos, such as the the Lyapunov dimension of strange attractors (Spratt, 2003). For these reasons, the calculation of the Lyapunov spectrum has been and still is of special interest to all relevant studies. However, the same systems that require a numerical solution (Sec. 2.1) also require a method for the numerical computation of their Lyapunov exponents. The literature on such numerical methods is vast¹⁵ and dates at least back to the 1970s, with the seminal article by Oseledets (1968) and the method developed in Benettin et al. (1980a, 1980b) which is considered the *standard*. The Lyapunov exponents of the systems presented and discussed in Sec. 3.3 were calculated using the Julia software library for chaos and nonlinear dynamics called *DynamicalSystems.jl* (Datseris, 2018), which implements the method proposed in Benettin et al. (1980b), as presented in Geist et al. (1990).

2.5 Attracting Sets, Attractors & Basins of Attraction

The asymptotically stable equilibrium solutions introduced in Sec. 2.3, also called sinks, are an example of what is known as an *attractor*. Informally speaking, an attractor is a subset of a dynamical system's state space onto which state space volumes *contract*. The set of all ICs that evolve toward the attractor is known as its *basin of attraction*. In 3D, an attractor can be an *equilibrium point*, a *limit cycle*, a *torus*, or a *strange attractor* (see Table 2.2, Sec. 2.6). Lastly, it is important to note that the above informal definition of an attractor indicates that *attractors only appear in the state space of dissipative systems* (Spratt, 2003).

Attractors actually belong to a more general class of characteristic sets called *attracting sets*. In order to help distinguish between the two, it is preferable to work toward a formal definition of an attractor by first providing one for an attracting set.

- With the system of ODEs $\dot{x} = f(x)$ (Sys. 2.1) and its flow $\phi_t(x_0)$ into focus, a subset Λ of the system's state space is called an attracting set if it satisfies the following conditions (Guckenheimer & Holmes, 1983; Wiggins, 2003):

¹⁵The interested reader can find a thorough survey of the numerical computation of Lyapunov exponents in Skokos (2010).

1. Λ is closed and invariant¹⁶.
 2. There is some neighbourhood U of Λ such that for each $x \in U$, $\phi_t(x) \in U$ for all $t \geq 0$, i.e. U maps onto itself under the forward evolution of ϕ_t , and $\bigcap_{t>0} \phi_t(U) = \Lambda$.
- The *basin of attraction* or *domain* B of the attracting set Λ is the set $\bigcup_{t \leq 0} \phi_t(U)$, i.e. the set of all points that under the forward evolution of ϕ_t are captured by the attracting set (Cvitanovic et al., 2020; Guckenheimer & Holmes, 1983).

The motivation for introducing the concept of an attractor on top of that of an attracting set comes from the importance of being able to describe the set which state space states fall into as precisely as possible, or more specifically, as an *indecomposable entity*. The need for that arises once it is noted that there is nothing in the definition of an attracting set that prevents it from being a collection of sets that are attracting in their own right, in which case the original set is *decomposable*. An attracting set meets this added requirement when it is *topologically transitive*. The definition of topological transitivity is given below.

- A closed invariant set M is topologically transitive if for every pair of open sets U and V in M , there exists a non-negative time $t \in \mathbb{R}$ such that $\phi_t(U) \cap V \neq \emptyset$. What this means is that under the forward evolution of ϕ_t , every point in M eventually comes arbitrarily close to every other point in M (Hirsch et al., 2004; Wiggins, 2003).

Having provided all definitions prerequisite to that of an attractor, this characteristic set of a dynamical system's state space is defined, according to Wiggins (p. 110 2003), as follows:

- "An attractor is a topologically transitive attracting set".

2.6 Chaos & Strange or Chaotic Attractors

The introduction of the terms discussed in this section is almost always accompanied by the following "disclaimer": *There is no universally accepted definition of chaos or strange/chaotic attractors...* ". This however, implies that for each of those terms there exists one or more definitions, with regards to which the Nonlinear

¹⁶According to J.C. Sprott's qualitative definition, "a set is *invariant* if a trajectory that starts on it remains on it for all time" (p. 56 Sprott, 2003)

Dynamics community has not been able to reach a consensus. The reason for this, apart from it being a multifaceted and complex subject matter, is that depending on the research topic, every author delves into different aspects of it and oftentimes, different aspects imply different properties and justifiably so, different definitions. Furthermore, the latter two terms, namely *strange* and *chaotic attractor*, are often used interchangeably (Sprott, 2003), but not always, which means that one must stay alert while multisourcing.

Making a clear distinction between different definitions and choosing one over the other is of no particular interest in the present study, since it does not concern itself with identifying strange/chaotic attractors, or confirming their status as such. Strange/chaotic attractors however, are the central feature of the technology presented here, so the need to provide even a rough description of them cannot possibly be overlooked. For these reasons, the definitions and general material provided in this subsection aim at offering just enough information to enable the conceptualisation of those terms.

According to R.L. Devaney (Devaney, 1989, p. 50), a dynamical system with vector field $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, like [Sys. 2.1](#), is *chaotic* if

1. It has sensitive dependence on initial conditions (unpredictability).
2. It is topologically transitive (indecomposability).
3. Its periodic orbits are dense in \mathbb{R}^3 (element of regularity).

The only property of the above that has not been mentioned so far is the density of periodic points. As a subset of the system's state space \mathbb{R}^3 , the set of periodic points P is said to be dense in \mathbb{R}^3 , if its *closure* $\overline{P} = \mathbb{R}^3$ ¹⁷.

While the above definition is one of the most technical definitions of chaos, and certainly one that has sparked many formal discussions – there are quite a few articles dedicated to various aspects of it – the latter two properties of chaos it focuses on are not easily established for most systems of interest (Hirsch et al., 2004; Sprott, 2003). The sensitive dependence on initial conditions, however, is quantified by the Lyapunov exponents introduced in [Sec. 2.4](#). For this reason chaos is very often defined using more relaxed sets of properties, which do however, always include sensitive dependence on initial conditions. An example of this is the definition given in Sprott (2003, p. 104):

¹⁷The closure \overline{P} of set P is the set that includes P itself and all of P 's *limit points*. A limit point of a set $P \subset \mathbb{R}^3$ is a point x in \mathbb{R}^3 – but not necessarily in P – every neighbourhood of which contains a point in P other than x (in case $x \in P$).

- “Chaos is the aperiodic, long-term behavior of a bounded, deterministic system that exhibits sensitive dependence on initial conditions”.

In this definition a chaotic trajectory needs to be *aperiodic*, since periodic motion is of highest order – quite the opposite of chaotic motion, *bounded*, so that there is the necessary state space folding¹⁸ capable of creating fractals – objects closely related to chaotic motion (see below), and to have *sensitive dependence on initial conditions*, or state space stretching¹⁹, which causes exponential separation of initially nearby trajectories (Spratt, 2003).

The definitions of an attracting set and an attractor were given in Sec. 2.5. Strange/chaotic attractors, as their name suggests, are attractors, but endowed with additional properties. When the distinction between the two terms is important, it is generally said that the term *strange attractor* describes objects with a particular *geometric structure*, and the term *chaotic attractor* objects in which trajectories exhibit a particular *dynamical behaviour* (Ditto et al., 1990; Grebogi et al., 1984). Specifically,

- Strange attractors have *fractal structure* (Feudel et al., 2006), and
- Chaotic attractors have *sensitive dependence on initial conditions* (Ditto et al., 1990; Grebogi et al., 1984).

Even though there are attractors that are both strange and chaotic – in fact, all attractors used in this study fall into this category – there are also attractors that are strange but not chaotic and vice versa (Spratt, 2010, and references therein).

Determining whether an attractor is strange or not is most often done via its *fractal dimension* D_F , which has a *non-integer* value when the attractor is indeed strange. The fractal dimension usually used for this purpose is known as the *capacity* or *box-counting dimension* D_{Cap}^0 (Farmer et al., 1983), a member of a parametric family of dimensions called *generalised dimensions*, with (integer) parameter q . The parameter values of primary interest are 0, 1 and 2; the capacity dimension corresponds to $q = 0$, while $q = 1$ and $q = 2$ define the *information dimension* D_{Inf}^1 and *correlation dimension* D_{Corr}^2 respectively (Feudel et al., 2006).

¹⁸The general comment made in Sec. 2.1 about complexity in dynamical systems arising from nonlinearities can now be partly justified by noting that the presence of nonlinearities in the deterministic law of a system are necessary for folding to take place in state space (Spratt, 2010).

¹⁹A prototypical example of state space stretching and folding is the *Smale horseshoe* or *horseshoe map* (Smale, 1967); a map that produces a fractal set after repeated stretching and folding in state space.

Lastly, there is the *Kaplan-Yorke* or *Lyapunov dimension*, defined in terms of the Lyapunov exponents L_j , $j = 1 \dots, m (= 3)$ (Kaplan & Yorke, 1979) as

$$D_L = D + \frac{1}{|L_{D+1}|} \sum_{i=1}^D L_i, \quad (2.33)$$

where D is the the largest index j for which $\sum_{i=1}^j L_i \geq 0$ – which J.C. Sprott calls the attractor's *topological dimension* (Sprott, 2003, p. 121).

For systems with more than two dimensions, it has been conjectured that the Lyapunov dimension is equal to the information dimension D_{inf}^1 (Farmer et al., 1983; Frederickson et al., 1983; Kaplan & Yorke, 1979). Note that the generalised dimensions are ordered in descending order as

$$D_{Cap}^0 \geq D_{inf}^1 \geq D_{Corr}^2. \quad (2.34)$$

Since $D_F = D_{Cap}^0$, if $D_L = D_{inf}^1$ then $D_F \geq D_L$, which means that the Lyapunov dimension might be a lower bound for the fractal dimension. Since establishing the “strangeness” of an attractor is of no interest in this study, but calculating the Lyapunov exponents is, the Lyapunov dimension will be used as an “estimator” of the fractal dimension, to indicate that the attractors discussed in Sec. 3.3 are in fact strange.

In the subsection introducing the Lyapunov spectrum (Sec. 2.4), it was stated and demonstrated geometrically (Fig. 2.3) that the Lyapunov exponents L_j , $j = 1, \dots, m (= 3)$ are measures of the average growth, or shrinking, of state space volumes, in different directions. More importantly, it was also mentioned that the Lyapunov exponents quantify the sensitive dependence on initial conditions, thus enabling the detection of chaos. What this latter statement implies, without expressly stating it, is that for exponential separation of nearby trajectories to take place in any direction, *at least one* of the characteristic exponents must be *positive* (see Eqs. 2.22 – 2.24). Due to the convention that puts L_j in descending order (Eq. 2.25), that *one* Lyapunov exponent will necessarily be L_1 , which is why it was given the “special” name *largest* or *maximal Lyapunov exponent*. Since, based on the definition given above, identifying a chaotic attractor means establishing sensitive dependence on initial conditions, if upon calculation of the Lyapunov spectrum the maximal Lyapunov exponent is positive, the attractor is deemed chaotic.

Before closing this section on Nonlinear Dynamics and Chaos, it might be constructive to list the type of attractors a dissipative dynamical system can exhibit in 3D, especially as they relate to the signs of the Lyapunov exponents. J.C. Sprott did that in a very interesting way in (Sprott, 2010): The properties of the Lyapunov spectrum in 3D impose certain rules on the signs the three characteristic exponents can have, which limit them to *four* possible combinations. Each of those combinations corresponds to a different type of attractor. The properties of the Lyapunov spectrum in 3D are the following:

- If the trajectory is not a fixed point (stable, since it is an attractor),
 1. The Lyapunov exponents cannot all be negative – they are for the fixed point. This means that $L_1 \geq 0$.
 2. At least one Lyapunov exponent must be zero (Haken, 1983).
- The sum of the Lyapunov exponents L_j , $j = 1, 2, 3$ of a dissipative system must be negative²⁰. This means that $L_3 < 0$.

The four allowed combinations are given in Table 2.2, paired with the name of the attractor they correspond to and the type of motion taking place in them. The type of motion named *quasiperiodic* refers to trajectories that consist of two – in the 3D case – independent periodic motions, with *incommensurate frequencies*, i.e. frequencies whose ratio is an irrational number (Sprott, 2010). To indicate the presence of two incommensurate frequencies, this type of attractor is called a *2-torus*. A quasiperiodic trajectory winds around the surface of the torus and without ever intersecting itself, fills it completely. An attracting fixed point has three negative exponents. A limit cycle has a zero maximal exponent – it corresponds to the direction of the flow parallel to the cycle – and the other two are negative. A 2-torus has its two largest exponents equal to zero – they correspond to the two independent periodic motions – “*one the short way around the torus, an the other the long way*” (Sprott, 2010, p. 19) – and the last one negative. And finally, a chaotic attractor has a positive maximal exponent, one equal to zero and a negative one.

²⁰A heuristic argument for that links the rate of volume change dV/dt – known to be negative for a dissipative system (Sec. 2.2, Eq. 2.13) – to $\sum_j L_j$ by noting that the volume of an ellipsoid is proportional to the product of its semi-axes, or, using the analysis made in Sec. 2.4, $V \propto \prod_j \sigma_t^j$. The Lyapunov exponents are given by Eq. 2.32, which implies that $L_j \propto \ln \sigma_t^j$ and ultimately that V should contract as $e^{\sum_j L_j t}$ and $dV/dt \propto V \sum_j L_j$.

Table 2.2: Types of attractors in 3D dissipative systems, identified by the signs of their Lyapunov exponents: The left column gives the combination of signs of the three Lyapunov exponents – or their value, when it is zero – based on the properties of the Lyapunov spectrum discussed in the main text. The second column lists the names of the attractors based on the combination they correspond to, and the third column specifies the type of motion associated with each of the attractors.

| Lyapunov exponents | | | Attractor Name | Type of Motion |
|--------------------|-------|-------|----------------|----------------|
| L_1 | L_2 | L_3 | | |
| – | – | – | Fixed Point | Stationary |
| 0 | – | – | Limit Cycle | Periodic |
| 0 | 0 | – | 2-Torus | Quasiperiodic |
| + | 0 | – | Chaotic | Chaotic |

2.7 Chaotic Synchronisation & Homogeneous Driving

Synchronisation is a universal phenomenon observed in a wide variety of *non-linear, dissipative* dynamical systems. Synchronisation generally refers to the *adjustment of rhythms* occurring when two self-oscillating systems, i.e. systems capable of producing their own rhythms without any external forcing, interact with one another (Pikovsky et al., 2001). One of the first class of chaotic systems shown to have the ability to synchronise, were actually *subsystems* forming *composite systems* of higher dimensionality, known as *drive decomposable systems*. *Homogeneously driven systems* are a subset of the latter class of systems. The subsystems of homogeneously driven compound systems, exhibit one of the simplest types of synchronisation, which is the main topic of the present subsection.

The interaction between two (sub)systems that (a) are *unidirectionally coupled* by one or more of their components, meaning that the *coupling components* allow one system to affect, or *drive* the other, without itself being affected by the *driven system*, and (b) the driven system is *identical* to the *driving system's* components not involved in the coupling, is known as *homogeneous driving* (Pecora & Carroll, 1991). The driving system is called the *drive* (Pecora & Carroll, 1990), or, depending on the context, *transmitter* (Cuomo, 1994) or *master* (see Ramirez et al., 2020, and references therein), the driven system is called the *response*, *receiver*, or *slave* respectively, and the pair of them forms the homogeneously driven compound system mentioned above.

Pecora and Carroll (1991) mathematically described the above concepts by introducing a formulation which is best understood using a dynamical system of known, specific dimensions. Let $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ be an m -dimensional autonomous dynamical system like [Sys. 2.1](#), after omitting the parameter vector \mathbf{p}_S for notational simplicity. If $m = 3$, which is the dimensions of the system implemented in this thesis ([Fig. 2.1](#)), then consider [Sys. 2.1](#) in its expanded form of [Sys. 2.2](#), repeated here for convenience.

$$\begin{aligned}\dot{x}^1 &= f^1(x^1, x^2, x^3) \\ \dot{x}^2 &= f^2(x^1, x^2, x^3) \\ \dot{x}^3 &= f^3(x^1, x^2, x^3).\end{aligned}$$

The system formed by *duplicating* one, or two of the above equations, say the first and the third, and appending them to the above system after some change in notation whose purpose will be become clear below, is

$$\begin{aligned}\dot{x}^1 &= f^1(x^1, x^2, x^3) \\ \dot{x}^2 &= f^2(x^1, x^2, x^3) \\ \dot{x}^3 &= f^3(x^1, x^2, x^3) \\ \dot{\tilde{x}}^1 &= f^1(\tilde{x}^1, x^2, \tilde{x}^3) \\ \dot{\tilde{x}}^3 &= f^3(\tilde{x}^1, x^2, \tilde{x}^3).\end{aligned}\tag{2.35}$$

This is an $(m + 2)$ -dimensional compound system, whose subsystems are unidirectionally coupled by the x^2 component, which allows the top subsystem to drive the bottom subsystem, without being affected by its \tilde{x}^1 and \tilde{x}^3 components. Moreover, the bottom subsystem (response) is identical to the driving subsystem's components x^1 and x^3 , which are not involved in the coupling. Therefore, [Sys. 2.35](#), is a homogeneously driven, *and* drive decomposable system. Note, however, that the above construction of a drive decomposable system cannot be applied on any system like [Sys. 2.1](#), since not all vector field components f^i depend on all state variables, which means that certain components cannot always drive the *duplicates* of others. This points will be made clear in [Subsec. 3.3.7](#), where the Lorenz system will be used as the basis for homogeneously driven composite systems.

Pecora and Carroll (1990) showed that, when the *response subsystem* of a drive decomposable compound system is *stable*, the two subsystems will *synchronise*. The stability of the response subsystem is confirmed when all of its Lyapunov exponents (Sec. 2.4), which in this case are called *partial* or *conditional Lyapunov exponents*, are *negative*. Synchronisation, in this context, means that starting from different ICs, after a certain period of time, the \tilde{x}^1 and \tilde{x}^3 components of the response subsystem will coincide with the respective components of the drive. The above concepts will be further elucidated in Subsec. 3.3.7.

Part I

Methodology:

Design & Software Architecture

Fairy tales are more than true: not because they tell us that dragons exist, but because they tell us dragons can be beaten.

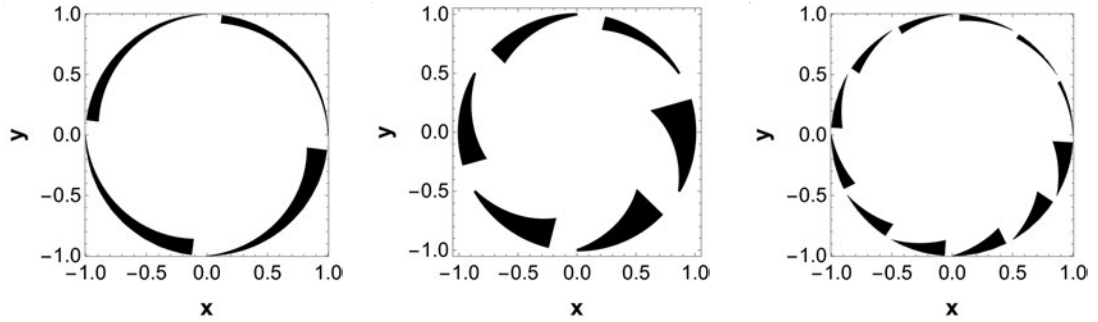
NEIL GAIMAN,
citing G.K. CHESTERTON
(Chesterton, 2007; Gaiman & McKean, 2012)

3

Structural Elements of the NLCode

3.1 Arced Circular Frame

The 2D pattern of the NLCode is enclosed in an *arcid circular frame* which defines the functional area of the feature. As Fig. 3.1 suggests, this frame can, in principle, consist of any number n_A of arcs. The frame's arcs begin with a certain width – the same for all arcs – which increases along the arcs such that each successive arc, except for the first one, ends wider than the one preceding it.



(a) 4-arcid frame starting thin ($\ell_{TH} = 10^{-3}$) at 0 rad angle. (b) 6-arcid frame starting thick ($\ell_{TH} = 10^{-2}$) at $\pi/6$ rad angle. (c) 12-arcid frame starting thin ($\ell_{TH} = 10^{-3}$) at 0 rad angle.

Figure 3.1: NLCode's arcid circular frame for three different sets of parameters. The complete set of parameters that specify the frame is given in Table 3.1 at the end of this section, and is illustrated in Fig. 3.2. In the frames shown here, the parameters varied are the number of arcs n_A , the polar angle θ_{init}^1 at which the first arc begins, the angles θ_A and θ_{GAP} spanned by a single arc and the blank space separating two successive arcs respectively, the thickness factor ℓ_{TH} of the plotted lines, and the width of the end of the last arc W_{LA} . All these quantities are discussed in detail in the main text.

The two geometrical components of the NLCode, i.e. the 2D encoded pattern in the center of the feature and the arcid circular frame placed at a certain radial distance from the pattern, define three separate areas on the NLCode (see Fig. 3.2). Specifically, if the *outer* radius of the entire feature is denoted by R_{outer} , then let

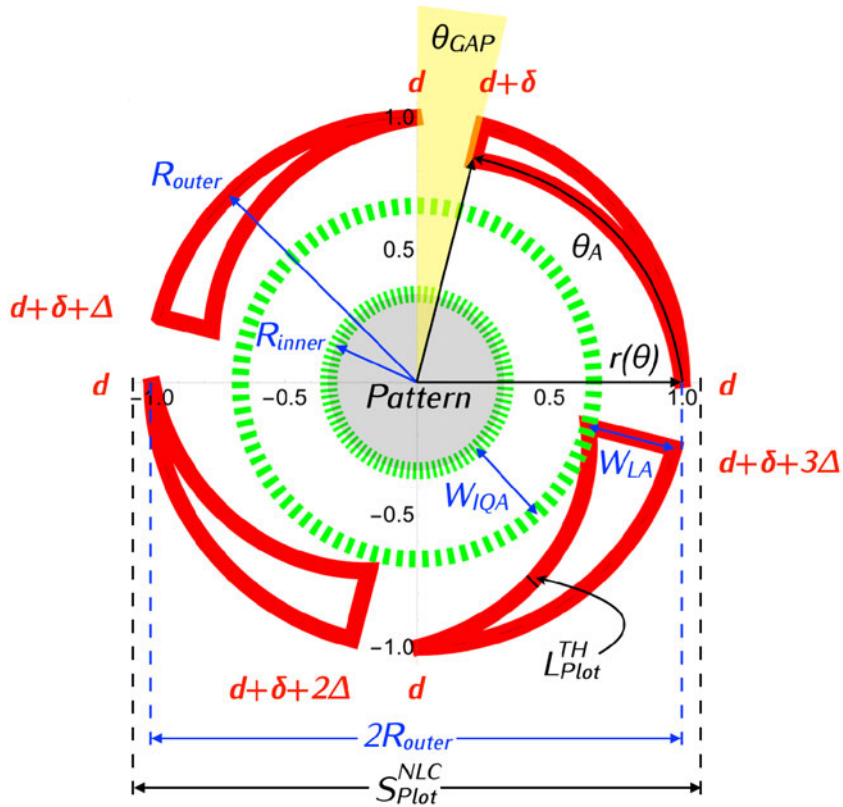


Figure 3.2: Detailed view of the arced circular frame's structure. Apart from introducing the parameters involved in the specification of the arced frame (also see Table 3.1), this illustration aims to highlight the technical aspects of the frame's design and provide visual aid in the clarification of certain implications involved in its creation using a computer software. For this purpose, line widths, circular areas and other features of the frame are presented in disproportion to one another and greatly exaggerated in size. In this figure one can see the three areas of the NLCode, namely the central area defined by R_{inner} which contains the 2D encoded pattern (inner green dashed circle), the inner quiet area of width W_{IQA} , and separated from the latter via a larger green dashed circle, the outer ring of width W_{LA} , enclosed within a circle of radius R_{outer} which contains the arcs. The scheme for the determination of the widths at the beginning d and the end $d + \delta + (i - 1)\Delta$ of the i -th arc is indicated around the circumference of the frame in red. In this particular instance the first arc starts at zero polar angle θ_{init}^1 and spans an angle of θ_A rad, followed by the blank space that spans θ_{GAP} rad and separates the first from the second arc. The inner edges of the frame's arcs are plotted using the parametric equations of a circle given by Eqs. 3.4, with variable radius $r = r(\theta)$ (Eq. 3.5). Lastly, the plot range S_{Plot}^{NLC} is defined by Eq. 3.10, i.e. with an added term dependent on the thickness L_{Plot}^{TH} of the lines drawn, in order to prevent erroneous clippings caused by the finiteness of the width of those lines (see main text for more details).

- W_{LA} be the width of the outer ring²¹ occupied by the frame, i.e. the width of the end of the *last arc* of the frame which is also the widest,
- W_{IQA} the width of the ring defining the *inner quiet area* of the NLCode, i.e. the blank space separating the frame and the encoded pattern, which ensures that during the decoding of the NLCode one is not mistaken for the other, and
- R_{inner} the *inner* radius of the disc containing the nonlinear pattern.

The last three quantities W_{LA} , W_{IQA} and R_{inner} can now be defined in terms of R_{outer} as follows:

$$W_{LA} = f_{LA} R_{outer} \quad (3.1a)$$

$$W_{IQA} = f_{IQA} R_{outer} \quad (3.1b)$$

$$R_{inner} = f_{inner} R_{outer} , \quad (3.1c)$$

with the non-zero $f_{LA} < 1$, $f_{IQA} < 1$ and $f_{inner} < 1$ representing the fraction of R_{outer} each of the three quantities is defined as. It becomes obvious upon a quick inspection of Fig. 3.2 that $W_{LA} + W_{IQA} + R_{inner} = R_{outer}$ which leads to the relation

$$f_{LA} + f_{IQA} + f_{inner} = 1 . \quad (3.2)$$

The beginnings and the endings of the arcs are determined as follows: The width all arcs begin with is denoted by d . The end of each arc is then increased by $\delta > 0$ ensuring that each arc will end wider than it began. This makes the end of every arc equal to $d + \delta$. By adding $(i - 1)\Delta$ to the latter sum, with $i = 1, \dots, n_A$ representing the i -th arc, it is ensured that, for some suitable value of parameter Δ – to be determined below, each successive arc except for the first one will end wider than the one preceding it, according to the requirement earlier specified. Figure 3.2 complements the above description by indicating the terms and quantities just introduced. From all the above, the width at the end of the last arc W_{LA} is seen to also be given as

$$W_{LA} = d + \delta + (n_A - 1)\Delta . \quad (3.3)$$

The inner edges of the frame's arcs are plotted using the parametric equations of a circle centred at (c_x, c_y) , with a variable (decreasing) radius $r = r(\theta^i)$, that is,

$$\begin{aligned} x^i &= c_x + r(\theta^i) \cos \theta^i \\ y^i &= c_y + r(\theta^i) \sin \theta^i , \quad i = 1, \dots, n_A . \end{aligned} \quad (3.4)$$

²¹In mathematical terms, the region between two concentric circles, here referred to as *ring*, is called an *annulus*, which in Latin means *little ring*.

The polar angle θ^i in the above equations takes values from an interval $[\theta_{init}^i, \theta_{fin}^i]$ which is different for each arc, but such that $\theta_{fin}^i - \theta_{init}^i = \theta_A$ for all $i = 1, \dots, n_A$, where θ_A is the polar angle spanned by a single arc. Considering that the variable radius r forming the inner edge of the i -th arc must be $r(\theta_{init}^i) = R_{outer} - d$ at the arc's beginning and $r(\theta_{fin}^i) = R_{outer} - [d + \delta + (i - 1)\Delta]$ at its ending, one easily deduces that

$$r(\theta^i) = R_{outer} - d - \frac{\delta + (i - 1)\Delta}{\theta_A} (\theta^i - \theta_{init}^i). \quad (3.5)$$

Another element of the arced frame, present through its absence of colour, is the space between the end of each arc and the beginning of the one succeeding it, at distance R_{outer} from the center, which will be referred to as *gap* (see Fig. 3.2). The gaps are initially defined as circular arcs whose length is a fraction/multiple of W_{LA} , i.e. as $f_{GAP} W_{LA}$, where $f_{GAP} \leq 1$. However, the quantity of immediate use is the angle θ_{GAP} spanned by the gaps, which at a radius R_{outer} is

$$\theta_{GAP} = f_{GAP} W_{LA} / R_{outer}. \quad (3.6)$$

Note that, just as is the case for the factors f_{LA} , f_{IQA} and f_{inner} , the number of arcs n_A and the angles θ_A and θ_{GAP} are not independent from one another, since

$$n_A (\theta_A + \theta_{GAP}) = 2\pi. \quad (3.7)$$

After specifying the number of arcs n_A , it is preferable to choose θ_{GAP} and let θ_A be determined through Eq. 3.7, since the separation of the arcs plays a far more important role in the functionality of the frame than the span of the arcs.

Before moving on to a brief study of the range of values of the parameters δ and Δ , it might be beneficial to discuss a few important factors involved in the creation of the frame, which relate to certain plotting specifications and quite possibly the particular software used to render the final image.

The software used to create and render NLCode samples is *Mathematica 12.3.1.0 Student Edition*. According to its documentation, the *thickness* of the lines drawn with Mathematica can be specified in one of two ways: (i) In an *absolute* manner, independent of the size of the final image, using units of printer's points, and (ii) As a fraction ℓ_{TH} of the plot range S_{Plot}^{NLC} along the horizontal axis of the plot. Since resizing an image does not change the plot range of the graph but rather "*stretches*" or "*shrinks*" it to fit the new size, the line width

must change in order to remain the same fraction of the “stretched” (“shrank”) plot range. This second approach is more suitable for most applications, and is the one used throughout this study, exactly because it allows the thickness L_{Plot}^{TH} of the lines drawn to be scaled up or down following the resizing of the image. Based on the above description, L_{Plot}^{TH} is defined as

$$L_{Plot}^{TH} = \ell_{TH} S_{Plot}^{NLC}. \quad (3.8)$$

► **A comment on notation**

The rather heavy notation used for the line width L_{Plot}^{TH} and the plot range S_{Plot}^{NLC} above, is due to the fact that these two quantities, as well as a few more that will appear in later chapters, can be expressed with respect to different coordinate systems, or refer to different processes. This creates the need for a comprehensive notation that enables all relevant distinctions. The coordinates all quantities introduced in the present section will be called *plot coordinates* (Sec. 3.2), hence the *Plot* subscript. The subscript *Print* will be used in Sec. 4.4 to indicate “instances” of the same quantities associated with the NLCode’s printed version. The letter *el* in different fonts, e.g. L , ℓ , indicated by the super/subscript *TH*, is used to denote various instances of the thickness of the lines drawn, and the letter *S* stands for *size* and usually refers to the size of the entire NLCode along the horizontal or vertical dimension (its size is the same in both dimensions). S_{Plot}^{NLC} in this particular instance is referred to as the *plot range* along the horizontal axis of the plot, but it is also the size of the NLCode expressed in plot coordinates. ◀

There is one intrinsic artifact – if it can be called that – related to plotting lines of finite width. In the case of a circle, for example, one might set the radius equal to a dimensionless 1, but if the line width is, say 0.05, then the line forming the circle will extend by 0.05/2 inwards and outwards of 1 in most software. This can be seen in Fig. 3.2 where the line widths have been exaggerated in order to illustrate this point. As a result, in order to avoid erroneous clippings of the NLCode’s frame, one must account for this artifact when specifying the plot range. This is achieved by letting

$$S_{Plot}^{NLC} = 2 R_{outer} + f_{PR} L_{Plot}^{TH}, \quad (3.9)$$

where $f_{PR} \geq 1$ is a factor allowing the adjustment of the plot margins. By substi-

tuting L_{Plot}^{TH} from Eq. 3.8 into Eq. 3.9 one can obtain a relationship for S_{Plot}^{NLC} that does not involve L_{Plot}^{TH} , i.e.

$$S_{Plot}^{NLC} = \frac{2R_{outer}}{1 - f_{PR} \ell_{TH}}. \quad (3.10)$$

S_{Plot}^{NLC} is indeed the plot range of the NLCode, in the literal sense of the term. However, when plot ranges are given as arguments to plotting functions, they are given in the form of intervals along each dimension, that is, for a 2D plot the plot range would be specified as $[(x_{min}, x_{max}), (y_{min}, y_{max})]$ which, in the case of the NLCode means that $S_{Plot}^{NLC} = x_{max} - x_{min} (= y_{max} - y_{min})$. Given that the arced circular frame is centred at (c_x, c_y) , a plot range specification in the above form would be

$$\left[\left(c_x - \frac{S_{Plot}^{NLC}}{2}, c_x + \frac{S_{Plot}^{NLC}}{2} \right), \left(c_y - \frac{S_{Plot}^{NLC}}{2}, c_y + \frac{S_{Plot}^{NLC}}{2} \right) \right]. \quad (3.11)$$

The finiteness of the lines drawn obviously affects the width of the arcs as well. This means that, by giving d some specific value, the i -th arc actually plotted begins with width $d + L_{Plot}^{TH}$ and ends with width $d + \delta + (i - 1)\Delta + L_{Plot}^{TH}$ (Fig. 3.2). Setting d to zero remedies the situation for the beginnings of the arcs, but does little for their endings. Similar, but not as profound – or as easily quantifiable, is the effect on the gaps separating the arcs. Factor f_{GAP} can be used here to make the appropriate adjustments. In any case, as long as the effects of the finiteness of the plotted lines is acknowledged and taken into consideration, there is no need to hardwire them into the equations (other than Eq. 3.9), a measure which would make things needlessly complicated and prone to errors.

After introducing all parameters and definitions pertinent to the arced circular frame of the NLCode, the construction of a frame that is visually as well as functionally successful depends on the appropriate specification of δ and Δ which are related via Eq. 3.3. Parameter Δ 's lower bound can – arbitrarily yet reasonably, see Fig. 3.2 – be set to

$$\Delta_{min} = \frac{d + \delta}{2}. \quad (3.12)$$

Note that the above limit implies that it is best to specify δ before Δ . Solving Eq. 3.3 for δ and using the fact that $\delta > 0$, leads to an “exclusive maximum” for Δ

$$\Delta_{max}^{exc} = \frac{W_{LA} - d}{n_A - 1} > \Delta. \quad (3.13)$$

The “exclusive minimum”²² – value of δ is known to be

$$\delta_{min}^{exc} = 0 < \delta. \quad (3.14)$$

A maximum value for δ is obtained by first noting that, for prespecified W_{LA} and n_A , if $\delta = \delta_{max}$ in Eq. 3.3, then Δ must equal Δ_{min} . Since Δ_{min} , through Eq. 3.12, is determined by whatever value δ is given, which in this case is δ_{max} , $\Delta_{min} = (d + \delta_{max})/2$. The maximum value of δ is obtained after these substitutions into Eq. 3.3, which allow it to be solved for δ_{max} giving

$$\delta_{max} = \frac{2 W_{LA}}{n_A + 1} - d. \quad (3.15)$$

Returning to a comment made earlier – that it is best to specify δ before Δ – and as an example, a choice that is simple and seems to work quite well is to set $\delta = \delta_{max}$ and then calculate Δ using Eq. 3.3, i.e.

$$\Delta(\delta) = \frac{W_{LA} - d - \delta}{n_A - 1}. \quad (3.16)$$

Table 3.1 lists all the parameters involved in the specification of the arced circular frame of the NLCode. The order in which they are listed is that in which they need to be calculated in order to specify and plot the frame. Some of the values given, such as the center $(c_x, c_y) = (0, 0)$ and the outer radius $R_{outer} = 1$ of the frame, are fixed values in the sense that other features of the NLCode have been thought through and designed around them (see Subsec. 3.2.1). With the single exception of the thickness factor ℓ_{TH} , the rest of them are reasonable suggestions that have been confirmed through testing. The thickness factor and its suggested range of values are further discussed and partially derived respectively, as part of a parametric study regarding the operating window of the NLCode scheme (Sec. 4.4).

²²The term *exclusive* is used here somewhat abusively to indicate characteristic values – “exclusive minimum/maximum” that despite being set as lower/upper bounds of a parameter, remain themselves unattainable by that parameter.

Table 3.1: Parameters involved in the specification of the arced circular frame of the NLCODE, divided in two columns. The independent parameters are given with one or more of the values that have been seen to render a functional frame and the dependent parameters are accompanied by the equations needed to calculate them. No dependent parameter is listed in the right column unless all of its dependencies have been first specified in the left column. The range of suggested values for the thickness factor ℓ_{TH} given here, is derived in [Sec. 4.4](#).

| Independent Parameters | Dependent Parameters |
|--|--|
| $(c_x, c_y) = (0, 0)$ | (x^i, y^i) arc boundaries, Eqs. 3.4 |
| $R_{outer} = 1$ | |
| $f_{LA} = 1/6$ | $W_{LA} = 1/6,$ Eq. 3.1a |
| $f_{IQA} = 1/6$ | $W_{IQA} = 1/6,$ Eq. 3.1b |
| | $f_{inner} = 2/3,$ Eq. 3.2 |
| | $R_{inner} = 2/3,$ Eq. 3.1c |
| $\ell_{TH} = 10^{-3}$ (up to 10^{-2}) | |
| $f_{PR} = 5$ | $S_{Plot}^{NLC} = 2.01,$ Eq. 3.10 |
| | $L_{Plot}^{TH} = 2.01 \times 10^{-3},$ Eq. 3.8 |
| $n_A = 4$ | |
| $d = 0.0$ | $\delta_{max} = 1/15,$ Eq. 3.15 |
| | $\delta = \delta_{max},$ |
| | $\Delta = \Delta(\delta),$ Eq. 3.16 |
| $\theta_{init}^1 = 0$ (up to $\pi/2$) | |
| $f_{GAP} = 3/4$ | $\theta_{GAP} = 1/8,$ Eq. 3.6 |
| | $\theta_A = (4\pi - 1)/8,$ Eq. 3.7 |

3.2 Geometry of the Digital NLCode

The three main components of the NLCode are listed below.

- The arced circular frame, which has already been presented in detail in [Sec. 3.1](#).
- The nonlinear pattern of the feature, whose creation from a generating system is the subject of [Sec. 3.3](#).
- The colour profile of the NLCode, which is thoroughly discussed in [Sec. 3.4](#).

The geometry of the NLCode is mostly an attribute of the first two of its components, expressed in their geometrical properties. The equations that represent the circular frame and the system of ODEs that generates the NLCode pattern are each expressed with respect to a certain frame of reference, and the two are not the same, not even in dimensions; the frame is 2D, while the dynamical system's state space is 3D. Moreover, when all is said and done and a digital NLCode is created and ready for use, it will be in the form of a digital image, which is accessed and navigated using pixel positions.

The 2D reference frame of the NLCode's arced circular frame will be referred to as *plot coordinate system*, due to the fact that the circular frame sets up the coordinate system for the plotting of the entire NLCode ([Subsec. 3.2.1](#)). The coordinates in the 3D reference frame of the dynamical system will be called *system coordinates*, and the coordinates of the 2D reference frame with respect to which digital images are processed *pixel coordinates*, both for obvious reasons.

In order to discuss the transformation between plot coordinates in 2D and the system coordinates in 3D without conflicting issues, one must first eliminate the dimensional mismatch between them. To that end, one can rather abstractly consider the system of plot coordinates as initially being a 3D system – and the circular frame of the NLCode a spherical shell, if one must – and then consider a 2D projection that creates the 2D plot coordinates and circular frame, in the same way as a 2D projection creates the NLcode's nonlinear pattern from the original 3D attractor. As a consequence of this mental exercise in rigour's honour, any reference to the plot coordinate system from this point onward will be accompanied by an explicit declaration of the dimensions implied, except in situations where the context allows no ambiguity.

As will be seen in [Subsec. 3.2.1](#), the transformation of system coordinates to 3D plot coordinates is very important for the systematic development of the NLCode,

and involves scaling and subsequently translating the system coordinates. In [Chapter 5](#) introducing the *NLCode Reader*, it will also become clear that the ability to transform back and forth between 2D plot and pixel coordinates is so crucial to the decoding of the NLCode, that it actually dictated most aspects of the feature's design. This transformation will be presented in [Subsec. 3.2.2](#).

3.2.1 Variable Transformation: System to Plot Coordinates

In many practical applications dealing with dynamical systems, it is often convenient, or necessary, to perform a transformation of variables in order to express the system of ODEs that describes the deterministic rule of the dynamical system, in terms of a new set of state variables. The new state variables will usually take values from a range more suited to the particular application. As an example consider an electronic implementation of [Sys. 2.1](#), for a specific f and known p_S . The dynamic range of the system variables, which in this case represent circuit voltages, cannot exceed the power supply limits²³. While in these situations it is usually sufficient to scale the system, for the application presented here it is required that the system be translated as well. The reasons for that are explained below.

The nonlinear pattern at the center of an NLCode is a strange and chaotic attractor like those discussed in [Sec. 2.6](#). This thesis will present two different dynamical systems in the form of [Sys. 2.1](#), whose attractor properties make them suitable for use in the creation of NLCode patterns. This is done in order to show that the NLCode scheme relies on certain general properties shared by many chaotic attractors, and can therefore perform equally well while displaying a variety of interesting looking patterns. The first system is discussed in [Sec. 3.3](#) and, in order not to disrupt the flow of this presentation by reporting results of the same nature twice, the study of the second system is provided in the [Appendix](#). Having a "library" of attractors to choose from, creates one issue: No two systems will ever exhibit an attractor of the same scale and at the same location in state space. This means that in order for the NLCode to be developed and applied in a systematic way, all available attractors must be brought to the same size and location. One function of the arced circular frame is to provide an appropriate coordinate system with respect to which an attractor can be scaled and translated.

²³Examples that also relate to the present work include (Cuomo & Oppenheim, 1993; Strogatz, 1994).

The general idea behind the scaling and translation applied to [Sys. 2.1](#) is to have the entire 3D pattern, e.g. the Lorenz attractor shown in [Fig. 2.1](#) (also see [footnote 8](#)), centred around a particular point in state space – the 3D version $\mathbf{c} = (c^1, c^2, c^3)^\top = (0, 0, 0)^\top$ of the circular frame's center $(c_x, c_y) = (0, 0)$, at a size just large or small enough to fit within a sphere of radius R_{inner} , which is no other than the radius of the central area of the NLCode dedicated to the encoded pattern (see [Table 3.1](#), [Sec. 3.1](#)).

Since the system is going to be subjected to two successive transformations, it is natural to wonder whether the order in which these transformations are performed matters or not. Before attempting to answer this question, one should bear in mind that, in general, the order of translation and scaling (and rotation, for that matter) of 3D objects does make a difference to the outcome of the entire operation as a whole. In this particular situation, if one were to first translate the 3D pattern's original center to the new center \mathbf{c} and then scale it according to some prescription, one would find that the subsequent scaling had displaced the center of the pattern away from its originally intended position, which is point \mathbf{c} . However, if the scaling is performed first, one need only mind that the subsequent translation moves the pattern's *new (scaled) center* – instead of its original center – to its intended position, which again is point \mathbf{c} . Based on the above reasoning, every system implemented in the present work will first be scaled and then have its scaled attractor's center²⁴ translated to the final point \mathbf{c} .

The system worked upon almost exclusively throughout this thesis is the fully transformed system, so it is preferable to reserve $\mathbf{x}(t) = (x^1(t), x^2(t), x^3(t))^\top$ for this system's state vector and assume that the original, untransformed system is

$$\dot{\boldsymbol{\chi}} = \boldsymbol{\varphi}(\boldsymbol{p}_S; \boldsymbol{\chi}) , \quad (3.17)$$

where $\boldsymbol{\chi} = \boldsymbol{\chi}(\tau)$ and τ denotes the time before any transformation of variables takes place (the scaling transformation can, optionally, change the time variable as well). The scaled system will be denoted by $\dot{\boldsymbol{X}} = \boldsymbol{F}(\boldsymbol{p}_S; \boldsymbol{X})$, where $\boldsymbol{X} = \boldsymbol{X}(t)$, and the scaled *and* translated system by $\dot{\mathbf{x}} = \mathbf{f}(\boldsymbol{p}_S; \mathbf{x})$, where $\mathbf{x} = \mathbf{x}(t)$.

In order to formulate the two transformations in a concise and uniform way, all vectors and matrices will be extended in dimensions by 2 – just for the calculations related to this transformation. This is done in two steps: In the first step the originally m -dimensional state vectors $\boldsymbol{\chi}$, \boldsymbol{X} and \mathbf{x} are added the extra

²⁴The center of an attractor like the Lorenz attractor shown in [Fig. 2.1](#) is defined in [Par. 3.2.1.1](#).

dimension of time and become $\chi(\tau) = (\chi^1, \chi^2, \chi^3, \tau)^\top$, $X(t) = (X^1, X^2, X^3, t)^\top$, and $x(t) = (x^1, x^2, x^3, t)^\top$ respectively. In the second step the same vectors are expressed in *homogeneous coordinates* (see e.g. Mortenson, 1999) and become $\chi(\tau) = (\chi^1, \chi^2, \chi^3, \tau, 1)^\top$, $X(t) = (X^1, X^2, X^3, t, 1)^\top$, and $x(t) = (x^1, x^2, x^3, t, 1)^\top$ respectively.

In the formulation of homogeneous coordinates, the scaling transformation is represented by the *scaling matrix* $S = \text{diag}(S^1, S^2, S^3, S^t, 1)$ and reads

$$X = S\chi \Rightarrow \begin{pmatrix} X^1 \\ X^2 \\ X^3 \\ t \\ 1 \end{pmatrix} = \begin{pmatrix} S^1 & 0 & 0 & 0 & 0 \\ 0 & S^2 & 0 & 0 & 0 \\ 0 & 0 & S^3 & 0 & 0 \\ 0 & 0 & 0 & S^t & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \chi^1 \\ \chi^2 \\ \chi^3 \\ \tau \\ 1 \end{pmatrix}, \quad (3.18)$$

where S^j , $j = 1, \dots, m (= 3)$ are the *spatial scaling parameters* and S^t the *temporal scaling parameter* such that $t = S^t \tau$. The method for calculating the spatial scaling parameters is presented in [Par. 3.2.1.1](#). Note that the requirement that the attractor is scaled simply to fit inside a sphere of a certain radius, implicitly assumes that this scaling is *uniform*, or, in other words, that the aspect ratio of the pattern is maintained. What this means is that the three spatial scaling parameters should be the same, i.e.

$$S^1 = S^2 = S^3 = 1/S, \quad (3.19)$$

where the inverted S , if anything, is a choice for intuition which may become clear in later paragraphs ([Subsec. 3.3.2](#) and [Sec. A.2](#)). The temporal scaling parameter S^t will be set equal to one, that is, $S^t = 1$, throughout this thesis.

The subsequent translation transformation is represented by the *translation matrix* \mathcal{T} and reads

$$x = \mathcal{T}X \Rightarrow \begin{pmatrix} x^1 \\ x^2 \\ x^3 \\ t \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & T^1 \\ 0 & 1 & 0 & 0 & T^2 \\ 0 & 0 & 1 & 0 & T^3 \\ 0 & 0 & 0 & 1 & T^t \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X^1 \\ X^2 \\ X^3 \\ t \\ 1 \end{pmatrix}, \quad (3.20)$$

where T^j , $j = 1, \dots, m (= 3)$ are the *spatial translation parameters* and T^t the *temporal translation parameter*. The latter is always assumed to be zero, i.e. $T^t = 0$,

because the dynamical systems treated here are autonomous and a translation in time is not expected to make a difference. The spatial translation parameters are calculated as follows: If $\chi_c = (\chi_c^1, \chi_c^2, \chi_c^3, 0, 1)^\top$ is the original center of the 3D pattern, then after the scaling its new center will be, according to Eq. 3.18, $X_c = S\chi_c$. With this in mind for later use, the translation of a pattern centred around X_c to the new center c is expressed by

$$x = X + (c - X_c), \quad (3.21)$$

where $c = (c^1, c^2, c^3, 0, 1)^\top$ and $X_c = (X_c^1, X_c^2, X_c^3, 0, 1)^\top$. Comparing the above equation with Eq. 3.20 leads to the conclusion that the spatial translation parameters are

$$T^j = c^j - S^j \chi_c^j, \quad j = 1, \dots, m (= 3). \quad (3.22)$$

The one-step transformation that achieves the initial goal set is found by substituting X from Eq. 3.18 into Eq. 3.20. This gives

$$x = T S \chi \Rightarrow \begin{pmatrix} x^1 \\ x^2 \\ x^3 \\ t \\ 1 \end{pmatrix} = \begin{pmatrix} S^1 & 0 & 0 & 0 & T^1 \\ 0 & S^2 & 0 & 0 & T^2 \\ 0 & 0 & S^3 & 0 & T^3 \\ 0 & 0 & 0 & S^t & T^t \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \chi^1 \\ \chi^2 \\ \chi^3 \\ \tau \\ 1 \end{pmatrix}. \quad (3.23)$$

Equivalently, if X is substituted into Eq. 3.21, the one-step transformation is expressed in the concise form

$$x = S\chi + (c - S\chi_c). \quad (3.24)$$

In order for this transformation to be complete it needs to be differentiated with respect to time, so that it can be applied to both sides of a system of ODEs such as Sys. 2.1. By noting once more that $X = X(t)$ while $\chi = \chi(\tau)$, Eq. 3.18 is differentiated with respect to time t giving

$$\dot{X} = \frac{1}{S^t} S \dot{\chi}, \quad (3.25)$$

where $\dot{X} = dX/dt = (\dot{X}^1, \dot{X}^2, \dot{X}^3, 1, 0)^\top$ and $\dot{\chi} = d\chi/d\tau = (\dot{\chi}^1, \dot{\chi}^2, \dot{\chi}^3, 1, 0)^\top$. Similarly, differentiating Eq. 3.20 (or Eq. 3.21) with respect to time t gives

$$\dot{x} = \dot{X}, \quad (3.26)$$

so that the overall transformation for the time derivatives is

$$\dot{\mathbf{x}} = \frac{1}{S^t} \mathcal{S} \dot{\mathbf{X}}. \quad (3.27)$$

Note that solving [Eq. 3.24](#) and [Eq. 3.27](#) for \mathbf{x} and $\dot{\mathbf{x}}$ respectively is quite easy in this case, since the inverse of scaling matrix \mathcal{S} is simply a new diagonal matrix whose non-zero elements are the reciprocals of the original matrix's corresponding elements, i.e. $\mathcal{S}^{-1} = \text{diag}(1/S^1, 1/S^2, 1/S^3, 1/S^t) = \text{diag}(S, S, S, 1/S^t)$.

3.2.1.1 Scaling & Translation Specifications

Having introduced and thoroughly discussed the NLCode's arced circular frame in [Sec. 3.1](#), and in anticipation of applying to the dynamical systems of [Sec. 3.3](#) the transformation from system to 3D plot coordinates just discussed, it is necessary to introduce the simple methods used to calculate the center \mathbf{x}_c of the 3D attractor exhibited by the original, untransformed, system $\dot{\mathbf{X}} = \boldsymbol{\varphi}(\mathbf{p}_S; \mathbf{X})$ ([Eq. 3.17](#)), as well as the elements of the scaling matrix \mathcal{S} which, after the additional requirement that the aspect ratio of the pattern be maintained – [Eq. 3.19](#), are reduced to the single parameter S .

As already mentioned in [Sec. 2.1](#), the numerical solution of a system of 1st order ODEs is a sequence $\{\mathbf{x}_n\}$, $n = 0, 1, \dots, N - 1$ of N points in state space. This is what is visualised in the 3D plot of [Fig. 2.1](#), and this is what has almost interchangeably been referred to as the system's attractor, the NLCode's nonlinear pattern, or simply the pattern. The *center* of this pattern is a geometrical concept subject to the definition ascribed to it, a fact that leaves one with a few options. For example, the pattern's center can be defined as the *mid-range* $(\mathbf{x}_{max} + \mathbf{x}_{min})/2$ of the sequence, or – and this is the option taken here – as the sequence's *arithmetic mean*, or *average* defined as

$$\mathbf{x}_c = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}_n. \quad (3.28)$$

Estimating the center of a particular system's attractor is done here by producing via the RK4 scheme a large sequence of points – anywhere from $N = 100,000$ to $500,000$ depending on the system (see [Sec. 3.3](#)) – and then apply [Eq. 3.28](#) to it.

The scaling parameter S is determined by first defining the *farthest point* χ_f of sequence $\{\chi_n\}$ from its center χ_c , and then demanding that the two points' scaled versions X_f and X_c respectively are at distance R_{inner} , that is,

$$|X_f - X_c| = R_{inner}. \quad (3.29)$$

X_f and X_c are given from Eq. 3.18 after substituting S^j , $j = 1, 2, 3$ with $1/S$ from Eq. 3.19. Letting $X_f = (1/S)\chi_f$, $X_c = (1/S)\chi_c$ and then solving Eq. 3.29 for S leads to

$$S = \frac{|\chi_f - \chi_c|}{R_{inner}}. \quad (3.30)$$

Note that χ_f is the farthest point from the center χ_c strictly in 3D and that its projection on any of the 2D planes $\chi^1\chi^2$, $\chi^1\chi^3$, and $\chi^2\chi^3$ will generally not be the farthest from the center's respective projection. This, however, does not affect the construction of the frame which is centred around (c_x, c_y) and has radius R_{inner} irrespective of the projection being used and the relative placement of χ_f in that projection. This point is demonstrated in Sec. 3.3 (Figs. 3.5 and A.2).

3.2.2 Variable Transformation: Plot to Pixel Coordinates

A digital image is a representation of a picture as a 2D array of values that correspond to colours. The elements of the array are called *pixels* and they are arranged in it just like the elements of a 2×2 matrix; in rows numbered from top to bottom, and in columns numbered from left to right. A pixel of the i th row and the j th column of an image is identified by the pair of indices (i, j) . These two indices are also the *pixel coordinates* treated in this subsection. Figure 3.3 shows the NLCode's frame – a very coarse raster image of it for illustration purposes – centred at the origin of the 2D plot coordinate system, overlaid with the axes of the pixel coordinate system whose origin is at the top-left corner of the figure.

After the first variable transformation – from system to 3D plot coordinates – all quantities characterising the geometry of the NLCode are expressed in plot coordinates (3D or 2D projected). However, this coordinate system belongs to the back-end of the feature. On the front-end, a finished NLCode is a digital image whose pixel locations are expressed in pixel coordinates. Any process hoping to utilise the rich dynamics of even a 2D projection of a chaotic attractor, which is exactly what the NLCode Reader does, will have to be able to transform back

and forth between 2D plot and pixel coordinates. Establishing the transformation between these two coordinate systems is the second, but probably the most important function of the arced circular frame. The method used for the definition of a *unique* transformation between plot and pixel coordinates is called *least squares adjustment*, and its following description is mostly based on Wolf et al. (2014).

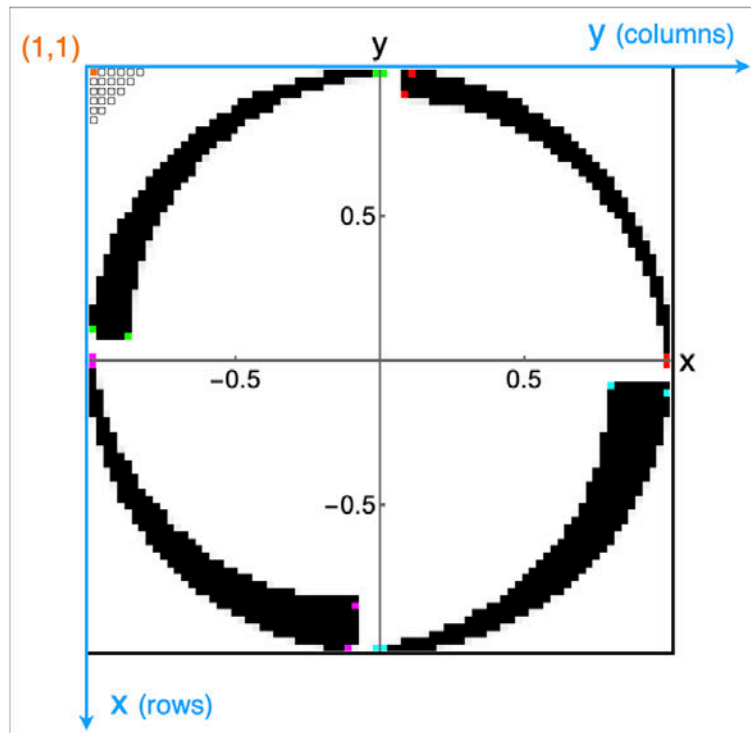


Figure 3.3: Transformation between plot and pixel coordinates, based on the arcs of NLCode’s circular frame. The plot coordinate system is shown in black and has its origin at the center of the frame. The pixel coordinate system is shown in light blue and its origin is at the top-left corner of the figure. The arced circular frame shown has $n_A = 4$ arcs and is very coarsely rasterised for illustration purposes. Its $3n_A$ corners are shown as colour-coded pixels, with the four colours red, green, magenta, and cyan, each used on the corners of a single arc. The transformation between the two coordinate systems is obtained via the method of least squares adjustment presented in this subsection. This method makes use of a redundancy in points whose coordinates are known in both systems. By design, in the case of the NLCode these points are the corners of the frame’s arcs, which can be located on a digital photograph of the NLCode using a corner detection algorithm (Subsec. 5.2.3).

Generally speaking, a transformation of variables is defined by a set of parameters, whose size n depends on the dimensions of the space which the transformation takes place in, and the type of the transformation. A quick inspection of the

two coordinate systems in Fig. 3.3 shows that the overall transformation between them, from either of the two systems to the other, should consist of a scaling, followed by a rotation of $\pm\pi/2$ rad about the origin, and finally a translation to bring the two origins together. This transformation belongs to a particular class of transformations called *affine transformations*. Assuming anisotropic scaling, i.e. a different scaling parameter in each dimension, this transformation introduces one scaling parameter per dimension, denoted by S^1 and S^2 as usual. The rotation transformation adds the rotation angle θ , and the translation introduces two more transformation parameters denoted by T^1 and T^2 . Therefore, in this case, the defining parameters of the transformation – and therefore the *unknowns* of the problem at hand – are *five*. However, for reasons that have to do with the formulation of the method employed, it will be assumed that they are six, i.e. $n = 6$.

Consider k points X_j , $j = 1, \dots, k$ on the NLCode, with known coordinates in both systems. The k pairs of corresponding sets of coordinates of those points in the two systems are the *input* to the method of least squares adjustment. Since the space in this case is 2D, i.e. $m = 2$, each pair of coordinates introduces two equations. In order to set up an algebraic system that will produce a unique solution, i.e. a unique transformation, the equations must be *at least* as many as the unknowns, that is, $mk \geq n$, or $k \geq 3$. Having said that, sometimes the available points exceed this minimum requirement, and $mk > n$. The least squares adjustment method was actually developed with the purpose to make use of this redundancy; it is a statistical method that utilises the additional information given by the excess pairs of coordinates, in order to also provide the standard deviations of the estimated transformation parameters.

Let $x_j = (x_j^1, x_j^2)$ represent one of the k points X_j in pixel coordinates, and $\chi_j = (\chi_j^1, \chi_j^2)$ the same point in 2D plot coordinates. The transformation between the two, described above as consisting of a scaling, a rotation and a translation, can be written as

$$\begin{aligned} x_j^1 + r_j^1 &= (S^1 \cos \theta) \chi_j^1 + (-S^2 \sin \theta) \chi_j^2 + T^1 \\ x_j^2 + r_j^2 &= (S^1 \sin \theta) \chi_j^1 + (S^2 \cos \theta) \chi_j^2 + T^2, \quad j = 1, \dots, k, \end{aligned} \quad (3.31)$$

where $r_j = (r_j^1, r_j^2)$ is the *residual* of x_j , which is defined below. In order to bring the above equations to a form that elucidates the workings of the method, the coefficients on their right hand side are assigned to a more uniform – notationally

wise – set of transformation parameters:

$$\begin{aligned} p_T^1 &= S^1 \cos \theta, & p_T^2 &= -S^2 \sin \theta, & p_T^3 &= T^1 \\ p_T^4 &= S^1 \sin \theta, & p_T^5 &= S^2 \cos \theta, & p_T^6 &= T^2. \end{aligned} \quad (3.32)$$

Notice that despite having only five proper transformation parameters, the coefficients of Eqs. 3.31 required $n = 6$ assignments, as anticipated. Of course, only five of those are independent.

Instead of just being vaguely “known” beforehand, let $x_j = (x_j^1, x_j^2)$ be an observation or measurement of X_j 's pixel coordinates, and note that in any practical implementation of the NLCode, every run of the NLCode Reader will produce one such measurement using a corner detection algorithm. With a sample of x_j 's measurements one can apply the well known formula of the arithmetic mean (Eq. 3.28) on each of the vector's components, to calculate what is also known as x_j 's *most probable value*, often denoted by \bar{x}_j . The (observation) residual is defined for each individual measurement of a quantity, as the difference between the observation and the quantity's most probable value, i.e., $r_j = x_j - \bar{x}_j$. This latter relation implies that when the transformation parameters are known and Eqs. 3.31 are used to transform a point's plot coordinates to pixel coordinates, the obtained quantity will actually be the *most probable value of the pixel coordinates*.

Having named one side of the known pair of coordinates *observations* (or *measurements*), let the other side, which is X_j 's plot coordinates, be referred to as *control coordinates* or more precisely *control points*, as is accustomed in studies implementing the method of least squares adjustment (Wolf et al., 2014).

Equations 3.31 were explicitly written for a single point X_j , but as the range of the j index implies, they apply to the entire set of the k points available. The algebraic system of mk equations can be written in matrix form as

$$\underbrace{\begin{pmatrix} x_1^1 \\ x_1^2 \\ x_2^1 \\ x_2^2 \\ \vdots \\ x_k^1 \\ x_k^2 \end{pmatrix}}_{\mathbf{X}[mk \times 1]} + \underbrace{\begin{pmatrix} r_1^1 \\ r_1^2 \\ r_2^1 \\ r_2^2 \\ \vdots \\ r_k^1 \\ r_k^2 \end{pmatrix}}_{\mathbf{R}[mk \times 1]} = \underbrace{\begin{pmatrix} X_1^1 & X_1^2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_1^1 & X_1^2 & 1 \\ X_2^1 & X_2^2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_2^1 & X_2^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_k^1 & X_k^2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_k^1 & X_k^2 & 1 \end{pmatrix}}_{\mathbf{X}[mk \times n]} \cdot \underbrace{\begin{pmatrix} p_T^1 \\ p_T^2 \\ p_T^3 \\ p_T^4 \\ p_T^5 \\ p_T^6 \end{pmatrix}}_{\mathbf{P}_T[n \times 1]} \quad (3.33)$$

or in the concise form

$$\mathbf{X} + \mathbf{R} = \mathcal{X} \cdot \mathbf{P}_T. \quad (3.34)$$

where \mathbf{X} is the *observation vector* – input data/pixel coordinates, \mathbf{R} the *residual vector* – observation residuals, \mathcal{X} is the *coefficient matrix* – input data/plot coordinates/control points, and \mathbf{P}_T the *vector of unknowns* – adjusted quantities/transformation parameters.

When the number of the observations (and control points) creates more equations than the number of unknowns, i.e. when $mk > n$, Sys. 3.34 is said to be *overdetermined*. In these cases, the objective of the least squares adjustment is not to simply find the solution of the system, but to optimise it so that it best fits the entire set of data (the k pairs of coordinates). This optimisation is also called *adjustment*, hence the term *adjusted quantities* in reference to the transformation parameters, used above.

The method achieves the adjustment of Sys. 3.34's solution by minimising the sum of the squared residuals. This process is best described using Eqs. 3.31 as a starting point; it involves solving each equation for the respective observation residual r_j^i , squaring both sides, and then summing the resulting equations to obtain an expression of the form

$$\sum_{j=1}^k \sum_{i=1}^{m(=2)} (r_j^i)^2 = G(\mathbf{x}, \mathcal{X}; p_T^\ell), \quad \begin{array}{l} i = 1, \dots, m(= 2) \\ j = 1, \dots, k \\ \ell = 1, \dots, n(= 6), \end{array} \quad (3.35)$$

where $G(\mathbf{x}, \mathcal{X}; p_T^\ell)$ is a scalar function of the adjusted quantities p_T^ℓ , parametrised by the observation and control data points. The above sum of squared residuals is minimised by taking its partial derivatives with respect to each of the adjusted quantities, p_T^ℓ , and setting them equal to zero. This step creates a new algebraic system of equations, called *normal equations*, of the form

$$\frac{\partial G(\mathbf{x}, \mathcal{X}; p_T^\ell)}{\partial p_T^\ell} = 0, \quad \ell = 1, \dots, n(= 6), \quad (3.36)$$

and its solution gives the optimised transformation parameters p_T^ℓ .

Returning to the matrix formulation which is what is used for the actual numerical calculations, the normal equations (Eq. 3.36) are written as

$$(\mathcal{X}^\top \mathcal{X}) \mathbf{P}_T = \mathcal{X}^\top \mathbf{X}, \quad (3.37)$$

and the final solution in matrix form is given by

$$\mathbf{P}_T = (\mathcal{X}^\top \mathcal{X})^{-1} \mathcal{X}^\top \mathbf{X}. \quad (3.38)$$

When there is no reason to assume that certain measurements are less, or more precise than others, which is the working hypothesis here, the method of least squares adjustment weighs all of them equally. The covariance matrix \mathcal{C}_T of the transformation parameters in this case, is given as (Ogundare, 2018)

$$\mathcal{C}_T = \sigma_0 (\mathcal{X}^\top \mathcal{X})^{-1}, \quad (3.39)$$

where \mathcal{X} is the coefficient matrix, and σ_0 is the *variance of an observation of unit weight* (Chilani, 2017), which is estimated by equation

$$\sigma_0 = \frac{\mathbf{R}^\top \mathbf{R}}{mk - n}, \quad (3.40)$$

where \mathbf{R} is the residual vector and $(mk - n)$ the number of adjustment equations minus the number of unknowns, which quantifies the *redundancy*, or *degrees of freedom* of a particular application of the method (Ogundare, 2018; Wolf et al., 2014).

The *standard deviations* $s_{\ell\ell}$ of the adjusted quantities are equal to the square roots of the variances $\sigma_{\ell\ell}$ of p_T^ℓ , which can be obtained from the diagonal of the covariance matrix, i.e.

$$s_{\ell\ell} = \sqrt{\mathcal{C}_T[\ell, \ell]}, \quad \ell = 1, \dots, n(=6), \quad (3.41)$$

where $\mathcal{C}_T[\ell, \ell]$ denotes the element in the ℓ th row and the ℓ th column of the covariance matrix \mathcal{C}_T .

Provided there exists a set of $k > 3$ points on the NLCode with known 2D plot coordinates, for which the pixel coordinates can somehow be found (see Par. 3.2.2.1), Eqs. 3.38 and 3.41 will give the transformation parameters that best fit these data pairs. The transformation *from plot to pixel coordinates* of any point X for which *only* the *homogeneous* 2D plot coordinates $\mathcal{X} = (\mathcal{X}^1, \mathcal{X}^2, 1)$ are known, is defined by the matrix equation

$$\begin{pmatrix} x^1 \\ x^2 \\ 1 \end{pmatrix} = \begin{pmatrix} p_T^1 & p_T^2 & p_T^3 \\ p_T^4 & p_T^5 & p_T^6 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathcal{X}^1 \\ \mathcal{X}^2 \\ 1 \end{pmatrix}. \quad (3.42)$$

3.2.2.1 Specification of Control Points and Observations

Just as the position and size of the NLCode's arced circular frame define the transformation between system and 3D plot coordinates, the arcs of the frame, and in particular the corners of the arcs, define the transformation between 2D plot and pixel coordinates.

Figure 3.3 of the previous subsection shows a pixelated frame with $n_A = 4$ arcs whose corners are indicated by colour-coded pixels: Red, green, magenta, and cyan – one colour for all three corners of each arc. In Sec. 3.1, the equations that form the frame's arcs were given in Cartesian and polar coordinates (Eqs. 3.4 and Eq. 3.5 respectively), both with respect to the 2D plot coordinate system. Reiterating the discussion around equation Eq. 3.5, the three corners of the i th arc are named, denoted and defined by the following equations, in polar coordinates:

$$\text{Beginning } A_B^i : (r_B^i, \theta_B^i) = [R_{outer}, \theta_{init}^1 + (i-1)(\theta_A + \theta_{GAP})] \quad (3.43a)$$

$$\text{Outer Ending } A_{oE}^i : (r_{oE}^i, \theta_{oE}^i) = [R_{outer}, \theta_{init}^1 + i\theta_A + (i-1)\theta_{GAP}] \quad (3.43b)$$

$$\text{Inner Ending } A_{iE}^i : (r_{iE}^i, \theta_{iE}^i) = [R_{outer} - (d + \delta + (i-1)\Delta), \theta_{init}^1 + i\theta_A + (i-1)\theta_{GAP}] \quad (3.43c)$$

If the frame can be plotted, all parameters entering the above equations are already specified (Table 3.1). This means that the plot coordinates of the $3n_A$ corners of the frame are known. These are the control points $\chi_j = (\chi_j^1, \chi_j^2)$, $j = 1, \dots, 3n_A$, that fill the coefficient matrix \mathcal{X} defined in Eq. 3.33, and they are obtained from Eqs. 3.4 after substituting the polar coordinates from the above Eqs. 3.43. What remains to be determined before the transformation parameters p_T^ℓ can be estimated using Eq. 3.38, is the set of *observations*, i.e. the pixel coordinates $x_j = (x_j^1, x_j^2)$ of the frame's corners.

In the final implementation of the NLCode, the NLCode Reader will process a photograph of the feature, taken by a smartphone camera. This is the situation of highest complexity faced by the developed technology, and is discussed in Chapter 5. What is treated in the present paragraph, is the simplest scenario possible, in which the following conditions apply:

- There is no noise present in the digital image of the frame, which suffers only from the artifact due to the finiteness of the line widths discussed at the end of Sec. 3.1.

- The image is a direct display of the frame's plot, with absolutely no perspective distortions.
- The image is not padded; there are no white pixels around the frame. This also means that the image is square, i.e. its size S_{Pixel}^{NLC} in pixels is the same in both dimensions.

The latter condition allows one to establish a correspondence between the size S_{Plot}^{NLC} of the NLCode in plot coordinates (Sec. 3.1), and the image size S_{Pixel}^{NLC} in pixels. What this means is that if one knows of a certain length, like a coordinate, in one of the two systems, one can apply the *rule of three* to express it with respect to the other coordinate system. In other words, this correspondence gives the scaling factor between the two systems which is the ratio $S_{Pixel}^{NLC}/S_{Plot}^{NLC}$ for the transformation from plot to pixel coordinates²⁵ (and its reciprocal for the inverse transformation). Note that, in the particular scenario examined, the scaling factor is the same in both dimensions.

As already stated in Subsec. 3.2.2, the transformation from plot to pixel coordinates also includes a rotation by $\pm\pi/2$ rad and a translation. Noting that coordinates transform opposite to their reference bases, which means that the transformation described here is the opposite of the transformation the plot coordinate axes would have to undergo in order to coincide with the pixel coordinate axes, leads to the following conclusions: (i) The plot coordinates need to be rotated about the origin by $+\pi/2$ rad (see Fig. 3.3). (ii) Given that the image is square and the NLCode frame is centred in it, the translation following the scaling and rotation, should be the same in both dimensions, with a translation parameter equal to $S_{Pixel}^{NLC}/2$.

Putting the three transformations just described into one matrix equation, the overall transformation from homogeneous plot to pixel coordinates reads

$$\begin{pmatrix} x_j^1 \\ x_j^2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \frac{S_{Pixel}^{NLC}}{2} \\ 0 & 1 & \frac{S_{Pixel}^{NLC}}{2} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos \frac{\pi}{2} & -\sin \frac{\pi}{2} & 0 \\ \sin \frac{\pi}{2} & \cos \frac{\pi}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{S_{Pixel}^{NLC}}{S_{Plot}^{NLC}} & 0 & 0 \\ 0 & \frac{S_{Pixel}^{NLC}}{S_{Plot}^{NLC}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_j^1 \\ x_j^2 \\ 1 \end{pmatrix}. \quad (3.44)$$

²⁵Typical pixel coordinates are of the order of 300 pixels, while plot coordinates, are of the order of $R_{outer} = 1$.

One very important remark about the transformation defined by Eq. 3.44 is that it appears to be exactly the transformation given by Eq. 3.42, that is, the transformation the entire Subsec. 3.2.2 was dedicated in finding (!) This is actually true, but only because of the special conditions listed in the beginning of this paragraph, under which this entire chapter operates. So in the present context, the transformation given above by Eq. 3.44 is all that is required by the NLCode Reader. However, in a practical implementation of the NLcode, it will be safer to use a corner detection algorithm in order to obtain actual observations of the corners of the arced frame, than to perform a series of corrections on the image – which will include corner detection anyway – and then exclusively rely on a single measurement of S_{Pixel}^{NLC} to set up the transformation of Eq. 3.44. In conclusion, the least squares adjustment method presented in Subsec. 3.2.2 is the preferred approach to estimating the transformation between plot and pixel coordinates in non-ideal conditions, based on observations of the arced frame's corners.

Having obtained the coordinates of the frame corners with respect to both systems, the observation vector X and the coefficient matrix \mathcal{X} are fully populated (Eq. 3.33), and Eq. 3.34 can be applied to give the vector of unknowns $P_T = (p_T^1, p_T^2, p_T^3, p_T^4, p_T^5, p_T^6)^T$. In order for the method to also provide the standard deviations of the adjusted quantities p_T^ℓ , the number of frame corners observed must be $k > n/m = 3$, out of the $3n_A$ in the frame.

3.3 Nonlinear Pattern: The Lorenz System

The Lorenz system describes a simplified model for atmospheric convection introduced by E.N. Lorenz in 1963 (E. Lorenz, 1963). It is the first system ever observed to possess what is today known as a *strange attractor* – a name first used in print by Ruelle and Takens (1971) – or *chaotic attractor* Sec. 2.6. For this reason, it is probably the best known, most extensively studied and widely used system in a variety of theoretical studies and engineering applications.

The Lorenz system is mathematically described by the following 3D system of 1st order ODEs

$$\begin{aligned}\dot{\chi}^1 &= \sigma(\chi^2 - \chi^1) \\ \dot{\chi}^2 &= \rho\chi^1 - \chi^2 - \chi^1\chi^3 \\ \dot{\chi}^3 &= \chi^1\chi^2 - \beta\chi^3,\end{aligned}\tag{3.45}$$

which – apart from the use of χ instead of x to indicate that the system has not yet been transformed – is of the general form of [Sys. 2.2](#). The system's state variables χ^1 , χ^2 and χ^3 represent the rate of convection and the horizontal and vertical temperature variation of the fluid Lorenz considered in his model respectively. The system parameters σ and ρ are the Prandtl and the Rayleigh numbers – both describing the relationship between certain fluid properties, and β relates to the physical dimensions of the system (Hirsch et al., 2004). All three of them are positive (Sparrow, 1982).

3.3.1 Numerical Solution

[Figure 3.4](#) shows a 3D trajectory of the Lorenz system ([Sys. 3.45](#)), solved using the RK4 method presented in [Sec. 2.1](#) with the original parameter values and ICs used by E. Lorenz (1963), i.e.

$$\sigma = 10, \quad \rho = 28, \quad \beta = 8/3 \quad (3.46a)$$

$$\chi_0^1 = 0.0, \quad \chi_0^2 = 1.0, \quad \chi_0^3 = 0.0. \quad (3.46b)$$

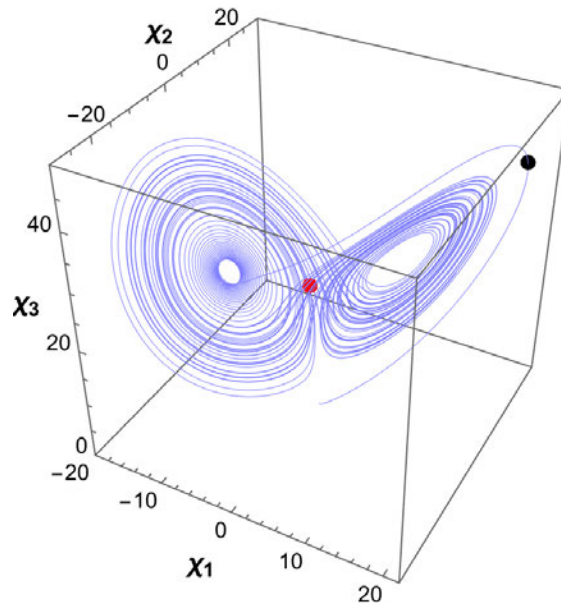


Figure 3.4: Lorenz attractor in 3D before the transformation: Solution of the Lorenz system ([Sys. 3.45](#)) using the RK4 scheme presented in [section 2.1](#). The parameter values and ICs are the ones originally used by E. Lorenz (1963) ([Eqs. 3.46](#)). The iteration step is $\Delta t = 10^{-3}$ seconds and the total number of iterations $N = 80,000$. The plot also shows in red the center χ_c of the attractor, calculated using [Eq. 3.28](#), and in black the farthest point χ_f from the center defined in [Par. 3.2.1.1](#).

3.3.2 Variable Transformation

The transformed form of the Lorenz system is obtained after applying on [Sys. 3.45](#) the one-step transformation given by [Eq. 3.24](#) ([Eq. 3.27](#)), and significantly simplified by setting $S^1 = S^2 = S^3 = 1/S$ ([Eq. 3.19](#)):

$$\begin{aligned}\dot{x}^1 &= \frac{\sigma}{T} (\mathbb{X}^2 - \mathbb{X}^1) \\ \dot{x}^2 &= \frac{1}{T} (\rho \mathbb{X}^1 - \mathbb{X}^2 - S \mathbb{X}^1 \mathbb{X}^3) \\ \dot{x}^3 &= \frac{1}{T} (S \mathbb{X}^1 \mathbb{X}^2 - \beta \mathbb{X}^3),\end{aligned}\tag{3.47}$$

where

$$(\mathbb{X}^1, \mathbb{X}^2, \mathbb{X}^3) = \mathbb{X} = \mathbf{x} - \mathbf{c} + (\boldsymbol{\chi}_c/S)\tag{3.48}$$

is a “collective variable” introduced to further simplify the transformed system’s algebraic form by taking advantage of a repeating expression. Note that even though $\dot{\mathbb{X}}^j = \dot{x}^j$, the original state variable \mathbf{x} on the left hand side of [Sys. 3.47](#) was retained for clarity.

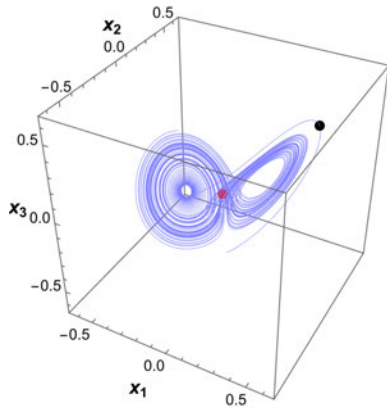
The calculation of the center $\boldsymbol{\chi}_c$ of the Lorenz attractor according to the definition given in [Par. 3.2.1.1](#) ([Eq. 3.28](#)), was done using a trajectory of $N = 500,000$ points. With $\boldsymbol{\chi}_c$ known, the computation of the farthest point $\boldsymbol{\chi}_f$ of the same trajectory from the center of the attractor was done simply by calculating the distance of every point in the trajectory from $\boldsymbol{\chi}_c$, finding the maximum distance, and matching it to a point which, evidently, is $\boldsymbol{\chi}_f$ ([Par. 3.2.1.1](#)). The results are given in the following set of equations and shown in [Fig. 3.4](#):

$$\boldsymbol{\chi}_c = (-0.5966, -0.5996, 23.5845)\tag{3.49a}$$

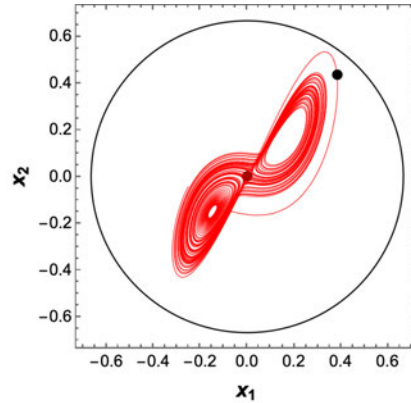
$$\boldsymbol{\chi}_f = (19.6953, 22.6041, 40.8823).\tag{3.49b}$$

The specification of the one-step transformation given by [Eq. 3.24](#) ([Eq. 3.27](#)) is complete as soon as the scaling parameter S is calculated using [Eq. 3.30](#), leading to

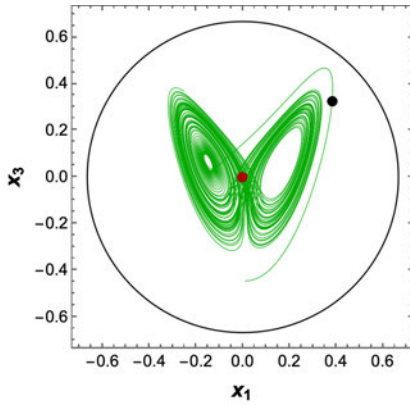
$$S = 53.02.\tag{3.50}$$



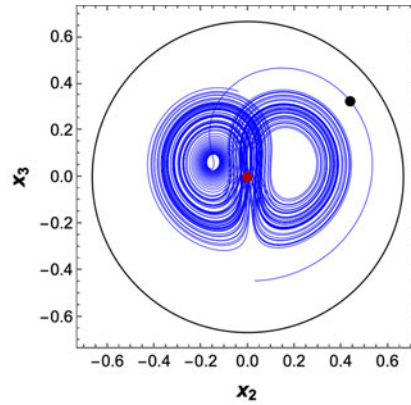
(a) Scaled and translated Lorenz attractor in 3D.



(b) Projection of the 3D trajectory on the x^1x^2 plane.



(c) Projection of the 3D trajectory on the x^1x^3 plane.



(d) Projection of the 3D trajectory on the x^2x^3 plane.

Figure 3.5: Lorenz attractor in 3D and 2D projections, after the transformation: (a) Solution of the transformed Lorenz system (Sys. 3.47) using the RK4 scheme presented in section Sec. 2.1. The parameter values are the same as before (Fig. 3.4), and the ICs are the transformed versions of the original ones given in Eq. 3.46b, specifically, $x_0 = (0.0113, 0.0302, -0.4448)$. The iteration step and the total number of iterations are the same as in Fig. 3.4. The plot also shows in red the transformed center $x_c = (0.0, 0.0, 0.0) = c$ of the attractor (Eq. 3.24), and in black the transformed farthest point $x_f = (0.3827, 0.4376, 0.3262)$ from the center. (b), (c), and (d) are the 2D projections of the trajectory and points x_c and x_f shown in Fig. 3.5a respectively. These plots also include the cross-sections (meridians) of the enclosing sphere of radius $R_{inner} = 2/3$ (Table 3.1) cut by the three planes. As discussed in the main text, the plots in this figure demonstrate the point that, while the two-dimensional projected x_f may not be farthest from the projected x_c , the cross-sectioned sphere by the respective plane will always encircle the attractor entirely – provided x_c and x_f were estimated with a sufficient accuracy to begin with, of course.

Figure 3.5 shows the Lorenz attractor after the scaling and translation transformations, and its three 2D projections on the x^1x^2 , x^1x^3 , and x^2x^3 planes. The scaled and translated versions \mathbf{x}_c and \mathbf{x}_f of the center and farthest point from center respectively are also shown in both 3D and 2D. These plots also demonstrate a point made earlier in Par. 3.2.1.1, that while \mathbf{x}_f is the farthest point from \mathbf{x}_c in 3D, its 2D projection will not necessarily be the farthest from the projected center. However, since the scaling is defined such that the transformed attractor is enclosed within a sphere of radius R_{inner} , any 2D projection of the attractor is bound to be contained in a circle of the same radius.

3.3.3 Dissipation

In order to determine whether the Lorenz system expands, preserves or contracts volumes in state space, one can use Eq. 2.14 (Sec. 2.2) to calculate the divergence of the system's vector field, or equivalently, the trace of its stability matrix. Taking the second route, the stability matrix of the transformed Lorenz system, Sys. 3.47, is found using Eq. 2.8,

$$\mathcal{A}(\mathbf{X}) = \frac{1}{T} \cdot \begin{pmatrix} -\sigma & \sigma & 0 \\ \rho - S\mathbb{X}^3 & -1 & -S\mathbb{X}^1 \\ S\mathbb{X}^2 & S\mathbb{X}^1 & -\beta \end{pmatrix}, \quad (3.51)$$

and its trace is then calculated in order to show, through Eq. 2.14, that

$$\nabla \cdot \mathbf{f} = -\frac{1}{T} (1 + \sigma + \beta) < 0 \quad (3.52)$$

which, according to the theory developed in Sec. 2.2 around Eq. 2.13, means that the Lorenz system is *dissipative*.

3.3.4 Equilibrium Solutions and Local Stability

For the set of system parameters used here (Eq. 3.46a), where most importantly $\rho > 1$, the Lorenz system has the three *real* equilibrium solutions given below. The fixed points \mathbf{x}_\ominus^* , \mathbf{x}_+^* and \mathbf{x}_-^* (the latter two jointly denoted by \mathbf{x}_\pm^*) are the

solutions of Eq. 2.15, for $\mathbf{f}(\mathbf{x}^*)$ equal to the vector field defined by Sys. 3.47:

$$\mathbf{x}_{\odot}^* = \mathbf{c} - \frac{\mathbf{X}_c}{S} \quad (3.53a)$$

$$\mathbf{x}_{\pm}^* = \left(c^1 - \frac{X_c^1}{S} \pm \frac{\sqrt{\beta(\rho-1)}}{S}, \right. \\ \left. c^2 - \frac{X_c^2}{S} \pm \frac{\sqrt{\beta(\rho-1)}}{S}, \right. \\ \left. c^3 - \frac{X_c^3}{S} + \frac{\rho-1}{S} \right). \quad (3.53b)$$

In the case of the first fixed point – which, for the original, untransformed Lorenz system (Sys. 3.45) is the origin $\mathbf{x}_{\odot}^* = \mathbf{0}$, hence the \odot subscript – the “collective variable” $\mathbf{X}_{\odot}^* = \mathbf{0}$ and the stability matrix (Eq. 3.51) becomes

$$\mathcal{A}(\mathbf{X}_{\odot}^*) = \frac{1}{T} \cdot \begin{pmatrix} -\sigma & \sigma & 0 \\ \rho & -1 & 0 \\ 0 & 0 & -\beta \end{pmatrix}. \quad (3.54)$$

The eigenvalues of $\mathcal{A}(\mathbf{X}_{\odot}^*)$ are found by solving Eq. 2.19:

$$\lambda_{\odot}^* = -\frac{\beta}{T} \quad (3.55a)$$

$$\lambda_{\odot\pm}^* = \frac{-(\sigma+1) \pm \sqrt{(\sigma+1)^2 + 4\sigma(\rho-1)}}{2T}. \quad (3.55b)$$

A first thing to note is that all three of the above eigenvalues are *real*. Secondly, $\lambda_{\odot}^* < 0$, $\lambda_{\odot-}^* < 0$, and for $\rho > 1$, which is the case treated here, $\lambda_{\odot+}^* > 0$. Thus, with “at least one eigenvalue with a positive real part, and one with a negative real part”, according to Sec. 2.2 (also see Table 2.1), \mathbf{x}_{\odot}^* is an *unstable* equilibrium solution of the *saddle* type.

Repeating the same process for the other two equilibrium solutions, that is, explicitly writing – or in some instances attempting to write – the analytical expressions of the stability matrix $\mathcal{A}(\mathbf{X}_{\pm}^*)$ and its eigenvalues, will turn out to be quite cumbersome and of little use to the essence of this presentation. For this reason, these results – including the first fixed point – are provided in numerical form, i.e. after substituting the parameter values of Eq. 3.46a into all relevant equations, in Table 3.2. The software used to perform all necessary calculations is *Mathematica 12.3.1.0 Student Edition*.

Table 3.2: Stability of the three equilibrium solutions of the Lorenz system. The first column lists the fixed points of *Sys. 3.47*, given by *Eqs. 3.53*, after substituting the parameter values from *Eq. 3.46a*. The second column lists the eigenvalues of the stability matrix calculated at the respective fixed points (entries of the same row). In the case of x_{\odot}^* , the eigenvalues were found from *Eqs. 3.55*, whereas in the case of the other two fixed points, the parameter values were first introduced into the computed stability matrix $\mathcal{A}(x_{\pm}^*)$, in order to then calculate its eigenvalues. Finally, the third column lists the stability type of the fixed points, based on the signs of the real parts of the eigenvalues, and according to the criteria presented in *Sec. 2.2* and summarised in *Table 2.1*.

| Equilibrium Point x^* | Eigenvalues of $\mathcal{A}(x^*)$ | Stability Type |
|---|--|----------------------|
| $x_{\odot}^* = (0.0113, 0.0113, -0.4448)$ | $\lambda_{\odot}^* = -2.6667$ $\lambda_{\odot+}^* = 11.8277$ $\lambda_{\odot-}^* = -22.8277$ | Saddle (unstable) |
| $x_{+}^* = (0.1713, 0.1713, 0.0644)$ | $\lambda_{\pm}^* = -13.8546$ | |
| $x_{-}^* = (-0.1488, -0.1487, 0.0644)$ | $\lambda_{\pm/\pm}^* = 0.0940 \pm i 10.1945$ | |

3.3.5 Lyapunov Spectrum & Lyapunov Dimension

The Lyapunov spectrum of the Lorenz attractor was calculated using the Julia software library *DynamicalSystems.jl* cited in *Sec. 2.4*, and was found to be

$$L_1 = 0.905, \quad L_2 = 0.0, \quad L_3 = -14.569. \quad (3.56)$$

Using the above results, the Lyapunov dimension is calculated from *Eq. 2.33* and turns out to be

$$D_L = 2.062. \quad (3.57)$$

With D_L a non-integer and one positive, one zero and one negative Lyapunov exponent (see *Table 2.2*, *Sec. 2.6*), the Lorenz attractor is confirmed to be both strange and chaotic, as expected.

3.3.6 Basin of Attraction

Figure 3.6 shows the basin of attraction (Sec. 2.5) of the Lorenz attractor. This set was computed using *DynamicalSystems.jl*, on a discrete 3D grid of size $(330 \times 300 \times 300)$, which means that it cannot fully represent continuous regions occupied by it. However, it is seen to fade with distance from the attractor, which indicates that even close to the attractor, there can be points which belong to neither of the two objects. A list of points belonging to the basin of attraction – which can, of course, be continuously updated – will later be used as a *pool of ICs* by the NLCode Generator (Chapter 4, Subsec. 4.2.1), in search of trajectories that satisfy a set of suitability criteria.

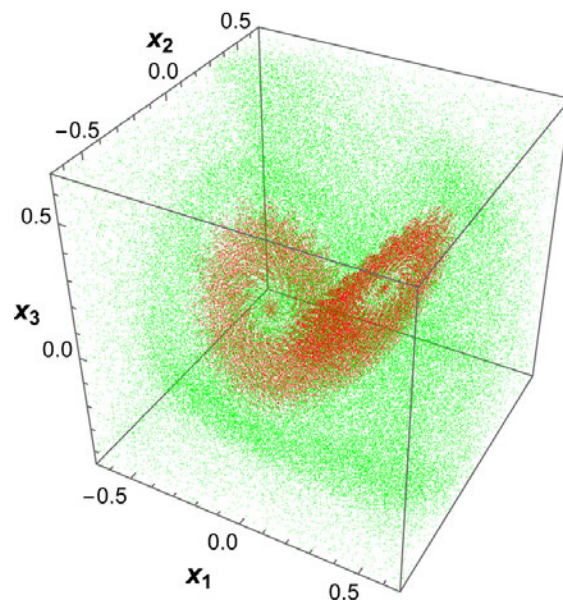


Figure 3.6: Basin of attraction of the Lorenz attractor. The Lorenz attractor is shown in red, and its basin of attraction in green. For the Lorenz attractor it is known that its basin of attraction extends to infinity (Sprott, 2003).

3.3.7 Homogeneous Driving & Synchronisation

According to what was presented in [Sec. 2.7](#), the *transformed* Lorenz system can be used as a base for the construction of two composite homogeneously driven systems. The first composite system is [Sys. 3.58](#). The coupling component in this case is x^2 of the drive subsystem, and the response subsystem consists of replicas of the first and third equations of the drive.

$$\begin{aligned}\dot{x}^1 &= \frac{\sigma}{T} (\mathbb{X}^2 - \mathbb{X}^1) \\ \dot{x}^2 &= \frac{1}{T} (\rho \mathbb{X}^1 - \mathbb{X}^2 - S \mathbb{X}^1 \mathbb{X}^3) \\ \dot{x}^3 &= \frac{1}{T} (S \mathbb{X}^1 \mathbb{X}^2 - \beta \mathbb{X}^3)\end{aligned}\tag{3.58}$$

$$\begin{aligned}\dot{\tilde{x}}^1 &= \frac{\sigma}{T} (\mathbb{X}^2 - \tilde{\mathbb{X}}^1) \\ \dot{\tilde{x}}^3 &= \frac{1}{T} (S \tilde{\mathbb{X}}^1 \mathbb{X}^2 - \beta \tilde{\mathbb{X}}^3).\end{aligned}$$

For the system parameters used in this implementation ([Eqs. 3.46](#)), the conditional Lyapunov exponents of this response system are both negative. Specifically,

$$L_1 = -10.000, \quad L_2 = -2.667.\tag{3.59}$$

The second composite homogeneously driven system is [Sys. 3.60](#), in which the coupling component is the drive's x^1 , and the response subsystem consists of replicas of the second and third equations of the drive.

$$\begin{aligned}\dot{x}^1 &= \frac{\sigma}{T} (\mathbb{X}^2 - \mathbb{X}^1) \\ \dot{x}^2 &= \frac{1}{T} (\rho \mathbb{X}^1 - \mathbb{X}^2 - S \mathbb{X}^1 \mathbb{X}^3) \\ \dot{x}^3 &= \frac{1}{T} (S \mathbb{X}^1 \mathbb{X}^2 - \beta \mathbb{X}^3)\end{aligned}\tag{3.60}$$

$$\begin{aligned}\dot{\tilde{x}}^2 &= \frac{1}{T} (\rho \mathbb{X}^1 - \tilde{\mathbb{X}}^2 - S \mathbb{X}^1 \tilde{\mathbb{X}}^3) \\ \dot{\tilde{x}}^3 &= \frac{1}{T} (S \mathbb{X}^1 \tilde{\mathbb{X}}^2 - \beta \tilde{\mathbb{X}}^3).\end{aligned}$$

The conditional Lyapunov exponents of this response system are also both negative, and specifically,

$$L_1 = -1.800, \quad L_2 = -1.866. \quad (3.61)$$

Since the response subsystems in both cases have negative conditional Lyapunov exponents, it is expected that starting with ICs different from their respective drives, after a certain period of time, the components of the response subsystems will coincide with those of the drives. As can be seen in Figs. 3.7 and 3.8, this is indeed the case.

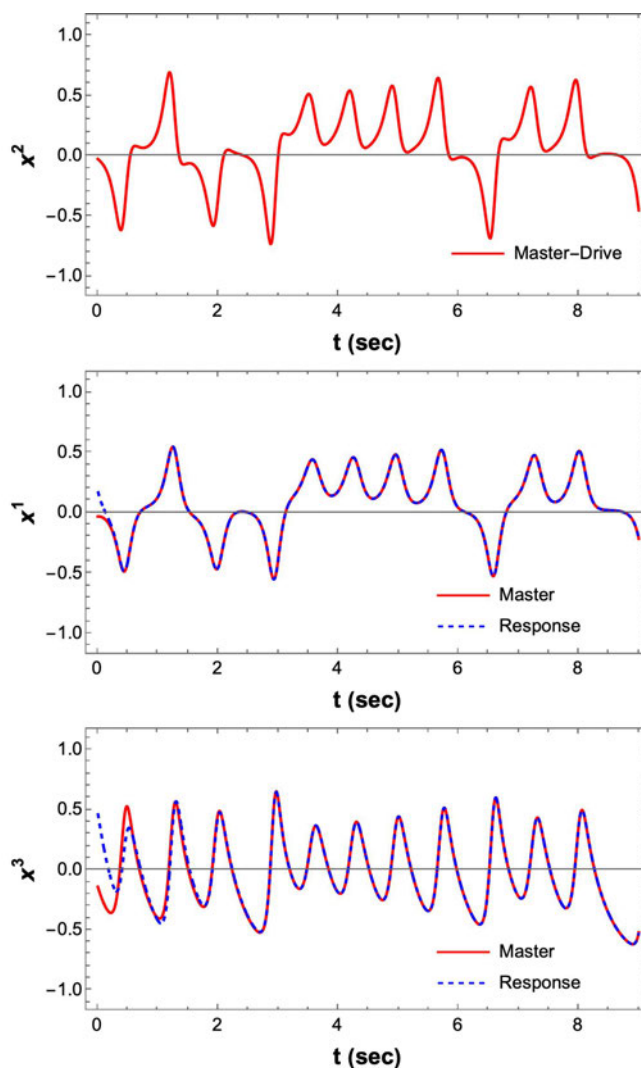


Figure 3.7: Synchronisation of (x^1, x^3) response, with x^2 drive: The top plot shows the coupling component. The other two plots show the corresponding components of the drive and response subsystems. Synchronisation occurs at some point after 2 sec.

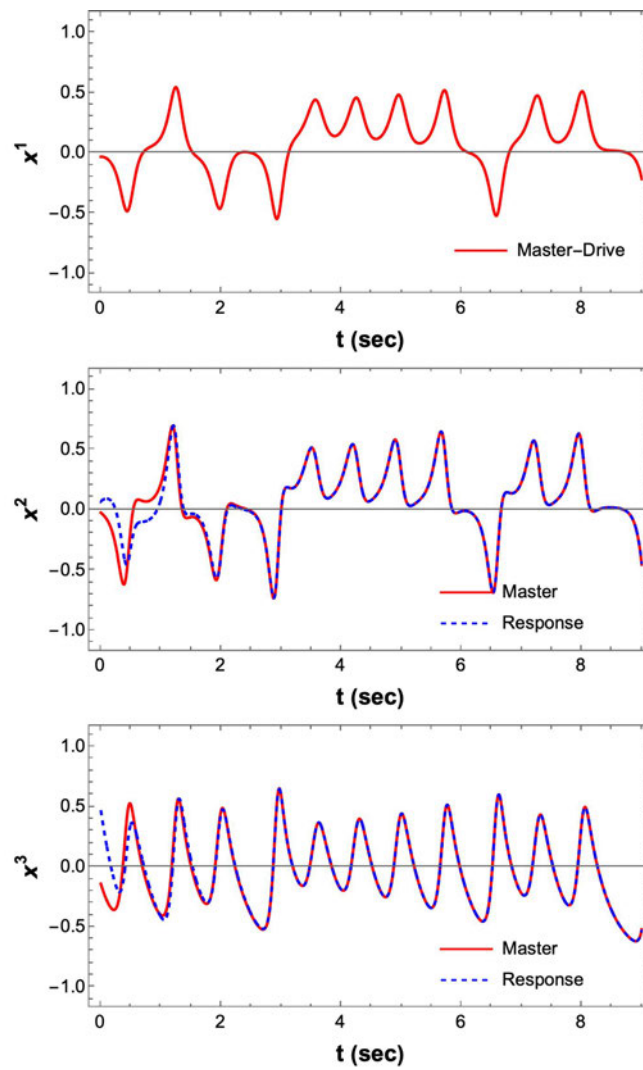


Figure 3.8: Synchronisation of (x^2, x^3) response, with x^1 drive: The top plot shows the coupling component. The other two plots show the corresponding components of the drive and response subsystems. Similarly to the previous case, synchronisation is seen to occur at some point after 2 sec.

3.4 Colour Profile of the NLCode

The colour profile of the NLCode describes the colour gradient of the plot line that forms the nonlinear encoded pattern. This feature of the NLCode serves two purposes: (i) It adds to the feature's aesthetic appeal considerably and by doing that, it strengthens the identifiability of the brand or branded product utilising the NLCode. (ii) With the right settings – determined by the NLCode Generator (Chapter 4) – the colouring of the nonlinear pattern acts as an auxiliary component to the main process implemented by the NLCode Reader, by making sure the latter “stays on track” in a both literal and metaphorical sense.

The colour profile of the NLCode is a sequence of colours creating a smooth gradient which is applied on the nonlinear pattern in a periodic fashion. It is defined by a sequence of elementary colours called *colour base*, a *sawtooth function* dictating the manner in which the colours of the base are interpolated in order to create smooth transitions from one colour to the next – a colour gradient effect, the *interpolation formula* itself, and the *period* of the sequence.

Let $\{i, j\}$, with $i, j = 1, 2, 3$ and $i \neq j$, denote the 2D projection of the strange attractor that forms the NLCode's nonlinear pattern, and assume that the projected trajectory consists of N points (x_n^i, x_n^j) , $n = 1, \dots, N$. According to the above description, the colour profile of the NLCode, denoted by \mathcal{C} , can be defined as the following sequence:

$$\mathcal{C} = \{c_1, c_2, \dots, c_N\} = \{c_n\}, \text{ with } c_{n+P} = c_n \text{ for all } n = 1, \dots, N, \quad (3.62)$$

where c_n are distinct colours, each assigned to its corresponding point (x_n^i, x_n^j) in the nonlinear pattern, and P the period of \mathcal{C} , which must be $P \leq N$. Note that the use of boldface fonts in Eq. 3.62, as well as in any other colour notation introduced in this section, makes sure that colours are treated as vectors, in anticipation of their mathematical representation using any of the colour models in wide use. Figure 3.9 illustrates an example colour sequence \mathcal{C} , for $N = 63$ and $P = 21$.

The construction of the NLCode's colour profile begins with the definition of its colour base. This is a sequence of colours denoted by \mathcal{B} , which has the form

$$\mathcal{B} = \{b_1, b_2, \dots, b_{N_{in}+1}\} = \{b_i\}, \quad i = 1, \dots, N_{in} + 1 \quad (3.63)$$

where b_i are $N_{in} + 1$ *base colours* with $b_{N_{in}+1} = b_1$, which means that only N_{in} of them are distinct. The reason for that is explained below.

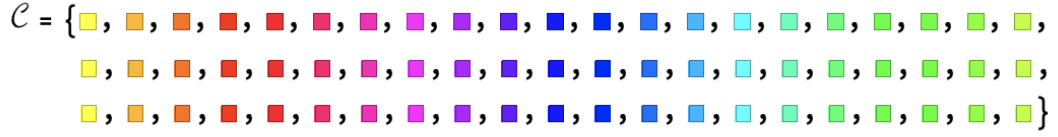


Figure 3.9: A sample periodic sequence of the NLCode's colour profile, for $N = 63$ and $P = 21$. Note that while this example is unrealistic in terms of the sequence's length and period – a typical 2D trajectory is expected to consist of 1000 to 8000 points – it highlights the two basic properties of \mathcal{C} , namely its periodicity and the smooth transition from one constituent colour to the next (gradient effect).

The next step is to introduce a function f of the index n of the colours in \mathcal{C} (Eq. 3.62), parametrised by the period P of the colour profile, i.e. $f = f(n; P)$. This function should be periodic with period P , and there should be an established correspondence between its values and the sequence of base colours, such that as the index n progresses, the base colours are periodically “selected” one by one from the beginning to the end of \mathcal{B} . In this current implementation of the NLCode, f is the *sawtooth function* defined as

$$f(n; P) = \frac{n-1}{P} \pmod{1}. \quad (3.64)$$

Figure 3.10 shows the plot of $f(n; P)$, for $P = 2$. Assuming a smooth increase of n from 1 to higher values, every time $n = sP + 1$, $s = 0, 1, 2, \dots$, f starts from 0 and linearly increases with n , approaching 1 as n 's increment from the previous integer value approaches P . It then sharply drops to 0 again, and the cycle repeats.

Since $f(n; P)$ varies from 0 to 1, the aforementioned correspondence between the function's values and the base colours \mathbf{b}_i is established by creating the pairs (β_i, \mathbf{b}_i) , where β_i are the $N_{in} + 1$ numbers that divide the interval $[0, 1]$ into N_{in} equal subintervals $[\beta_k, \beta_{k+1}]$, $k = 1, \dots, N_{in}$, i.e.

$$\beta_i = \frac{i-1}{N_{in}}, \quad i = 1, \dots, N_{in} + 1. \quad (3.65)$$

Let (c_n, \mathbf{c}_n) be a point with abscissa $c_n = f(n; P) \in [\beta_k, \beta_{k+1}]$, for one of k 's values. The colour of the n th point in the nonlinear pattern of the NLCode is defined as the above point's ordinate \mathbf{c}_n . With $f(n; P)$ known, the two points

(β_k, \mathbf{b}_k) and $(\beta_{k+1}, \mathbf{b}_{k+1})$ are also known, and \mathbf{c}_n can be calculated from the linear interpolation formula (see e.g. Hamming, 1973)

$$\mathbf{c}_n = \mathbf{b}_k + \frac{f(n; P) - \beta_k}{\beta_{k+1} - \beta_k} \cdot (\mathbf{b}_{k+1} - \mathbf{b}_k), \quad n = 1 \dots, N, \quad (3.66)$$

where once more k , indicates the subinterval $[\beta_k, \beta_{k+1}]$ which $f(n; P)$ belongs to.

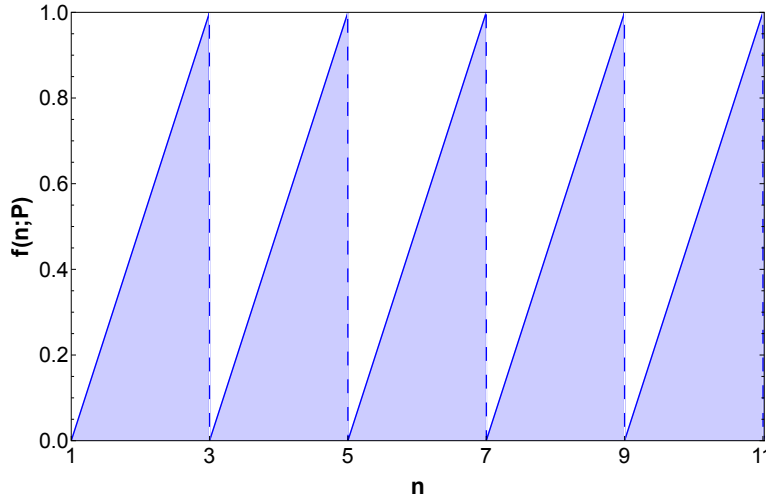


Figure 3.10: Plot of function $f(n; P)$ as defined by equation Eq. 3.64, for $P = 2$. This function belongs to a family of functions known as sawtooth functions, describing non-sinusoidal waveforms.

At the present stage of development, the NLCode has two colour bases. The first one uses a *colour theme* called *Hue*, which includes the primary additive and subtractive colours, in the order they appear on the colour wheel, i.e. yellow, red, magenta, blue, cyan, green, yellow; the repetition of yellow at the end of this colour sequence is a consequence of the definition of the colour base by Eq. 3.63, where $\mathbf{b}_{N_{in}+1} = \mathbf{b}_1$, and is explained below. This colour base is denoted by \mathcal{B}^{Hue} and its vector representation using the RGB colour model is

$$\mathcal{B}^{Hue} = \{(1, 1, 0), (1, 0, 0), (1, 0, 1), (0, 0, 1), (0, 1, 1), (0, 1, 0), (1, 1, 0)\}. \quad (3.67)$$

The second colour base is denoted by \mathcal{B}^{TM} , with *TM* abbreviating the name *Tin-masters* after this project's sponsoring company, because this base's theme consists of the colours that make up their brand's gradient, namely yellow, red, purple, yellow. This colour base's RGB vector representation is

$$\mathcal{B}^{TM} = \{(1, 1, 0), (1, 0, 0), (0.5, 0, 0.5), (1, 1, 0)\}. \quad (3.68)$$

Figure 3.11 shows the two colour bases of the NLCode's colour profile.



(a) The \mathcal{B}^{Hue} colour base (Eq. 3.67) includes the primary additive and subtractive colours as they appear on the colour wheel. For this base, the number of non-repeating colours is $N_{in} = 6$.



(b) The \mathcal{B}^{TM} colour base (Eq. 3.68) includes the colours that make up the Tinmasters' gradient, i.e. yellow, red and purple. For this base, the number of non-repeating colours is $N_{in} = 3$.

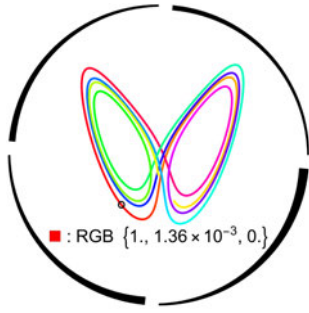
Figure 3.11: The two colour bases of the NLCode's colour profile. Note that while the number of non-repeating colours in these bases is N_{in} , their total length is $N_{in} + 1$. The repetition of the first colour (yellow) at the end of each sequence is explained in the main text.

The reasons for having the first colour (yellow) of the colour bases \mathcal{B}^{Hue} and \mathcal{B}^{TM} repeat at the end of these sequences are the following:

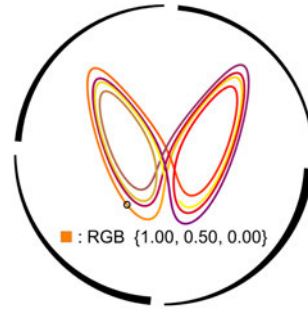
1. It ensures that the final colour sequence \mathcal{C} (Eq. 3.62) includes colours representing a smooth transition from green (purple) – the last non-repeating colour of each sequence, to yellow – its first (and last) colour, thus maintaining the gradient effect of the profile.
2. It makes sure none of the colours in the colour base is inadequately represented in the final sequence \mathcal{C} . The possibility of this happening stems from the fact that, depending on n 's increment (equal to 1 in NLCode's application) relative to the period P of the sawtooth function (Eq. 3.64), $f(n; P)$ may not approach 1 sufficiently to reproduce the last colour of the sequence. Repeating yellow at the end of the sequence \mathcal{B}^{Hue} (\mathcal{B}^{TM}), ensures that green (purple) – the last distinct colour in the base – is adequately represented.

Figure 3.12 demonstrates the effect of the relative values of P – the colour profile's period, and N – the number of points in the nonlinear pattern, on the appearance of the NLCode.

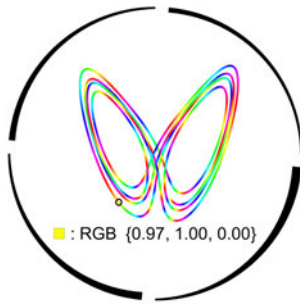
- In the first case examined, $P = N = 6000$, which means that the input base \mathcal{B} is spread over the entire pattern once, without repetition. This creates a slow, seemingly uneventful transition between colours, which unfortunately allows for overlaps of same colours at the line's crossings; *colour overlaps* are treated in Sec. 4.3.



(a) Colour base: \mathcal{B}^{Hue} , Period: $P = 6000$.
 \mathcal{B} is repeated exactly once.



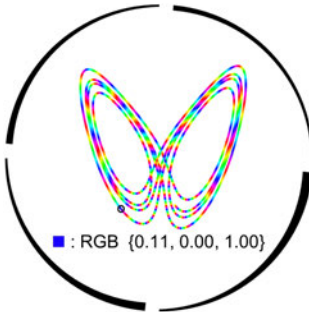
(b) Colour base: \mathcal{B}^{TM} , Period: $P = 6000$.
 \mathcal{B} is repeated exactly once.



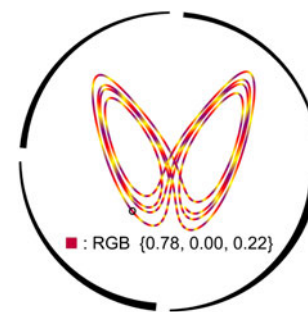
(c) Colour base: \mathcal{B}^{Hue} , Period: $P = 200$.
 \mathcal{B} is repeated 30 times.



(d) Colour base: \mathcal{B}^{TM} , Period: $P = 200$.
 \mathcal{B} is repeated 30 times.



(e) Colour base: \mathcal{B}^{Hue} , Period: $P = N/105$.
 \mathcal{B} is repeated 57.143 times.



(f) Colour base: \mathcal{B}^{TM} , Period: $P = N/105$.
 \mathcal{B} is repeated 105 times.

Figure 3.12: Application of the NLCode's colour profile that demonstrates the effect of the profile's period P relative to the size N of the 2D projected trajectory, on the appearance of the feature. All the NLCodes shown have the same nonlinear pattern which consists of $N = 6000$ points. The profiles on the left column all use the same colour base, \mathcal{B}^{Hue} , and vary the period P . The right column shows the effect of the same periods on the second colour base, \mathcal{B}^{TM} (Fig. 3.11). A black circle covers a small area around the $n = 1000$ th point of the nonlinear pattern, and its RGB colour – as calculated by the method presented in this section – is given in the legend of each figure. Further details are provided in the captions of the individual figures, and in the main text.

- The second period applied is $P = 200$, which divides $N = 6000$ equally. In this case, the colour base \mathcal{B} is repeated on the nonlinear pattern 30 times, which in effect, brings the colour transitions closer together and creates a more vivid picture.
- Lastly, in the third case examined $P = N/105$ with $N = 6000$, which makes $P = 57.143$ and shows first of all that P does not have to be in integer – in fact, the same applies to the number of \mathcal{B} 's repetitions over the nonlinear pattern. The colour transitions in this instance are quite fast, and that puts the functionality of the NLCode as well as its aesthetics in jeopardy.

The period of the NLCode's colour profile is one of the main adjustment parameters used by the NLCode Generator, in order to create nonlinear patterns which the NLCode Reader can process successfully. The precise function of this feature of the NLCode will be thoroughly discussed in [Sec. 4.3](#) and [Chapter 5](#).

4

The NLCode Generator

4.1 Objectives of the NLCode Generator

The nonlinear pattern of the NLCode is a trajectory of a 3D dynamical system like [Sys. 2.1](#), projected on the 2D planes x^1x^2 , x^1x^3 , and x^2x^3 (see e.g. [Fig. 3.5](#)). Generally speaking, the *area coverage* of a 2D projected trajectory and its *density* in state space – both qualitative measures, should be such that the NLCode has a pattern that is well positioned and distributed in the allocated area of the feature ([Sec. 3.1](#), [Fig. 3.2](#)). The *NLCode Generator* is the algorithmic process developed for the creation of nonlinear patterns with the above qualities. The reason it is sufficient for the desirable characteristics of suitable patterns to only be given a qualitative description, is that the NLCode Generator is mainly a *process of elimination*, i.e. it is designed to create patterns with certain qualities by rejecting those that have certain other, undesirable characteristics, at the same time advancing toward secondary, predefined goals. The need for a relatively complex process like the NLCode Generator to create readable NLCodes is illustrated in [Figure 4.1](#), which shows three not at all uncommon examples of nonlinear patterns with devastatingly poor area coverage and high density in state space.

In order to effect visible changes to the 2D trajectory, one can (i) alter the parameters of the system – without moving it out of the chaotic regime in which the system's behaviour is known, (ii) use a different set of ICs, (iii) *slightly* increase (or decrease) the number of points in the 2D trajectory, and (iv) alter the colour profile applied to the pattern by changing its period ([Sec. 3.4](#), [Eq. 3.64](#)). However, since these trajectories evolve in a chaotic attractor, which according to the discussion of [Sec. 2.6](#) is characterised by sensitive dependence on ICs, options (i)²⁶ and (ii) will not simply “*effect visible changes*” to a trajectory; they will in all likelihood change the entire trajectory. Concerning option (iii), note that this trajectory is aimed to be a *readable pattern*, which implies that at the very least, there have to

²⁶A chaotic system is equally sensitive to small deviations from an intended set of system parameters – again, within a certain chaotic regime – as it is to similar departures from an intended set of ICs. So in the present context, the term “*initial conditions*” generally refers to the *initialisation* of the system, which includes the system parameters *and* the actual starting point of the trajectory.

be discernible elements in it. This means that in order to prevent it from becoming too dense, the number of iterations used for its creation – forward and/or backward in time, if need be – should not exceed a certain limit. Finally, while the colour profile – option (iv) – can have a substantial visual effect on a pattern, it cannot affect its geometric characteristics. For this reason, alterations of the colour profile are only employed last, and only for patterns that have already been reviewed and accepted based on their spatial attributes.

In conclusion, “*slight adjustments*” in the geometry of the trajectories forming the nonlinear patterns of the NLCode are generally not possible. In most cases, if a trajectory fails the tests imposed by the NLCode Generator, there is little any reasonable adjustments to the number of iterations or the period of the colour profile can do; it has to be replaced by an entirely different trajectory. Once a trajectory meets certain geometrical criteria (see [Sec. 4.2](#)), it enters a second processing phase which is dedicated to colour profile adjustment ([Sec. 4.3](#)).

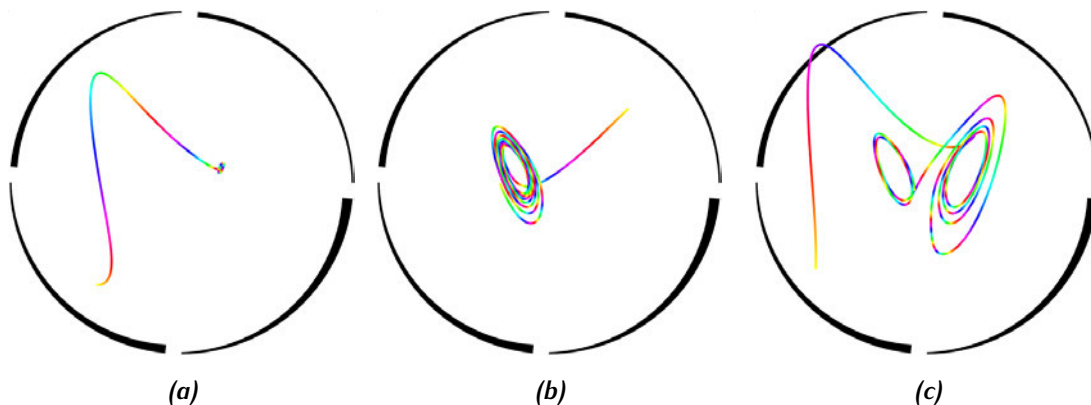


Figure 4.1: Patterns with poor area coverage and high density in state space. The example trajectories shown are fairly common occurrences in the search for suitable patterns by the NLCode Generator. One important note about these trajectories is that they were all produced by solving the Lorenz system for the same number of iterations $N = 5,000$. (a) 3D ICs: $(x_0^1, x_0^2, x_0^3) = (-0.4928, 0.0201, -0.5998)$. This trajectory started outside the attractor, but ended up revolving around one of the equilibrium points, x_+^* ([Table 3.2](#)). (b) 3D ICs: $(x_0^1, x_0^2, x_0^3) = (0.4615, -0.1538, 0.4303)$. Similar situation to (a), but with a wider and more visible twirl around x_-^* . (c) 3D ICs: $(x_0^1, x_0^2, x_0^3) = (-0.6533, -0.5864, -0.5017)$. This trajectory’s way to the attractor led it not only outside the pattern area of the NLCode, but outside the entire feature whose boundary is the arced frame²⁷. The rest of the parameters entering [Sys. 3.47](#) are $\mathbf{p} = (\sigma, \rho, \beta)$ from [Eq. 3.46a](#), $T = 1$, $S = 39.2741$, $\mathbf{c} = \mathbf{0}$, and χ_c from [Eq. 3.49a](#).

Figure 4.2 shows another example of a trajectory that failed to meet the criteria it was tested against during different phases of the Generator. This trajectory is well positioned and distributed in the pattern area and does not appear to be too dense to have different parts of it resolved. Even with a good coverage area and density, however, this trajectory failed not one, but all tests of the Generator. The figure points out and briefly describes the main undesirable attributes a trajectory might be afflicted by, that the Generator is built to identify and act against. Each of the characteristics descriptors provided corresponds to a task the Generator performs in order to ensure the viability of the NLCode. The remaining of this chapter breaks down the NLCode Generator to its constituent processes and discusses each of them in detail.

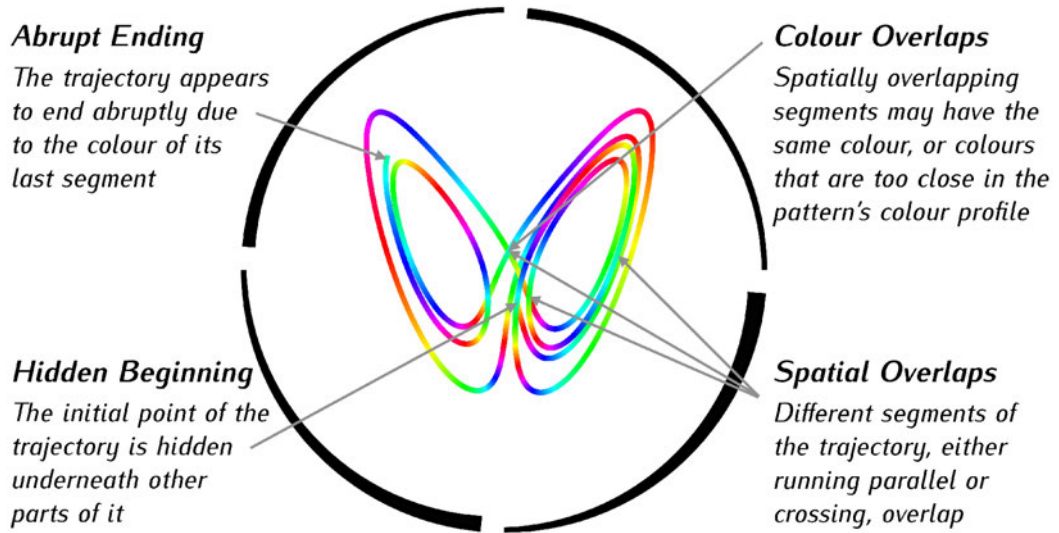


Figure 4.2: The four main tasks of the NLCode Generator correspond to undesirable characteristics of the trajectories created. The Generator is tasked to identify these inadequacies and act accordingly. The issues described as **Abrupt Ending**, **Hidden Beginning** and **Colour Overlaps** can be treated or even prevented, whereas the one described as **Spatial Overlaps** requires a change of trajectory, that is, a new set of ICs. Whatever the case is, the main challenge of the NLCode Generator is to accurately identify the culprits. Trajectory specifications: 3D ICs: $(x_0^1, x_0^2, x_0^3) = (0.0528, 0.0935, -0.1385)$, $N = 5,000$, $\mathbf{p} = (\sigma, \rho, \beta)$ from Eq. 3.46a, $T = 1$, $S = 30.2108$, $\mathbf{c} = \mathbf{0}$, and χ_c from Eq. 3.49a. Frame/Plot specifications: $R_{outer} = 1$, $\ell_{TH} = 10^{-2}$, and $f_{PR} = 3$ provide S_{Plot}^{NLC} from Eq. 3.10 and L_{Plot}^{TH} from Eq. 3.8.

²⁷A discussion regarding the optimal method for choosing ICs as well as the standard practice of discarding several points from the beginning of the trajectory in order to eliminate its transient part, are discussed in Subsec. 4.2.1.

4.2 Spatial Overlaps

The term *Spatial Overlaps (SOs)* refers to either one of two situations:

- **Self-crossings:** Different sections of a trajectory running *non-parallel* to one another pass through the same point, i.e. intersect. As already noted in [Sec. 2.1](#), trajectories in the original 3D state space can neither intersect with one another, nor self-cross. The self-crossings discussed here are a direct consequence of the projection of the 3D trajectory on the 2D planes. This effect is exacerbated as the number of iteration increases.
- **Contiguities:** In this case, different segments of a trajectory running *parallel*, or *almost parallel* to one another share several consecutive points, i.e. they are *contiguous*, or simply, they *touch*. The projection of the 3D trajectory in 2D is partly responsible for this type of SOs as well, but contiguities are also caused – even in 3D – by the *finite line width “artifact”* discussed in [Sec. 3.1 \(Eq. 3.9\)](#), i.e. the fact that while geometrical lines are one-dimensional, the lines drawn are inevitably two-dimensional. Increased line width and number of iterations unavoidably give rise to more contiguities, that also tend to extend to longer line segments.

SOs of the two types just described are an inevitable occurrence in any trajectory worth processing further. This means that the NLCode Generator needs some sort of quantifiable criteria based on which it can accept certain trajectories and reject others. This set of criteria and the methods for testing trajectories against them are presented in this section in the order in which they are applied to any given trajectory tested.

The first few steps of the NLCode generation process include the initialisation of the process, followed by the solution of the chosen dynamical system expressed in 3D plot coordinates using the RK4 method, and the creation of the arced circular frame. The latter also includes the calculation of the plot range S_{Plot}^{NLC} ([Sec. 3.1, Eq. 3.11](#)) required for the plotting of the two main geometrical components of the feature, thus allowing a completely unprocessed NLCode to be visualised from the early stages of its creation.

4.2.1 Initial Conditions & Transient Intervals

During the initialisation process of the NLCode Generator, the parameters that specify the main components of the feature are set according to a certain desired result. From that point onward, the Generator starts creating patterns based on these specifications, and puts them through a series of tests until it finds at least one that satisfies a set of criteria (Fig. 4.2). In order to do that, the Generator needs access to a large set of ICs, all of which create trajectories that evolve within the attractor of the system implemented. As was briefly mentioned in Subsec. 3.3.6, the *basin of attraction* of any given attractor provides an excellent *pool of ICs*, since all points in it will *eventually* be captured by the attractor (Sec. 2.5). The basin of attraction is also a *large set* – and can be added to or updated, which allows the Generator to pick ICs at random and also reuse the pool, since a trajectory rejected under one configuration might be suitable for another.

The implication of choosing ICs from the basin of attraction instead of the attractor itself, is that the trajectories created will only *eventually* start evolving within the attractor, that is, after a certain period of time. The parts of trajectories that correspond to these periods of time – or the time periods themselves, depending on the context – are called *transient intervals*. Figure 4.3 shows what a trajectory with ICs taken from the basin of attraction looks like including its transient interval, and what it looks like without it. Based on this figure, it becomes obvious that the NLCode Generator can only benefit from the use of basins of attraction as pools of ICs, provided it makes sure that the transient intervals are discarded prior to the processing of trajectories. Fortunately, this is as easy as taking this procedure into account when setting the total number of iterations N for the RK4 method, so that when the first N_{trans} iterations are discarded from the trajectory, its size is still the one intended. The number of iterations that correspond to transient intervals vary not only per system, but per trajectory as well. Provided the final trajectory is required to do a few “windings” inside the attractor, a good rule of thumb is to set N_{trans} to approximately a third of the total number of iterations required for those “turns” to take place. In the case of the Lorenz system, for example, $N = 3,000 - 5,000$ is a reasonable number of iterations for a trajectory. The Generator in this case may set $N_{trans} = 1,000$ and then reset N by adding N_{trans} to it in order to compensate for the iterations lost in the the omission of the transient interval.

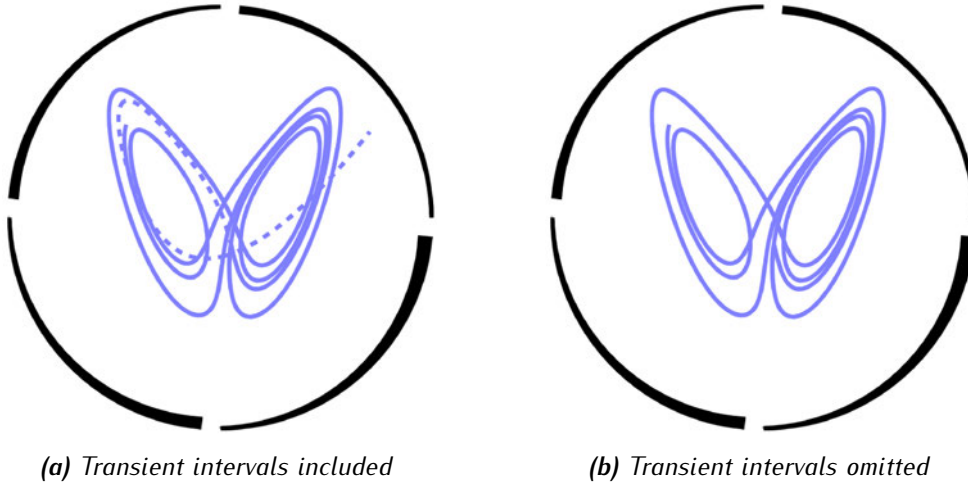


Figure 4.3: The effect of transient intervals on the NLCode pattern. When the ICs do not belong to the attractor of the system – as is the case for points taken from the attractor’s basin of attraction, trajectories spend a period of time outside the attractor before being captured by it. Those initial parts of the trajectories are called transient intervals and can have a devastating effect on the patterns created. As figure (a) on the left shows, a trajectory initiating from within the basin of attraction may have to cross the entire attractor – in the 2D projection – before it starts evolving in it. 3D ICs: $(x_0^1, x_0^2, x_0^3) = (0.7080, -0.6214, 0.4071)$. Figure (b) on the right shows the same trajectory with a transient interval of only $N_{trans} = 570$ iterations omitted, for demonstration purposes (new 3D ICs same as in Fig. 4.2). This is a necessary step the NLCode Generator must take before processing trajectories any further. The rest of the trajectory and frame/plot specifications are the same as in Fig. 4.2.

4.2.2 Sampling of the 2D Projected Trajectory

The NLCode Generator begins the processing of a candidate pattern by *sampling* the 2D projected trajectory, in preparation for the subsequent methods requiring it as input. Note that this kind of sampling is neither periodic, nor uniform in the time domain, as one might expect; the sampling performed by the NLCode Generator is based on spatial considerations. Specifically, if $x[n]$ is the initial, 2D projected trajectory and $x_s[n]$ the sampled trajectory, then

$$x_s[n] = \begin{cases} x[n], & n = 0 \\ x[n+k], & n \geq 0, k = k[n] \end{cases} : \|x_s[n+1] - x_s[n]\| \geq D \forall n, \quad (4.1)$$

where as before, $\|\cdot\|$ denotes the Euclidean vector norm. What the above expression means, is that the sampling of the 2D projected trajectory starts with the first

point $x[0]$, and each subsequent point is added in the sampled trajectory only if its Euclidean distance from the previous point that entered the sample is greater than or equal to some prespecified distance D – to be determined in the next subsection. Alternatively, one might say that the sampled trajectory is formed by taking every k points from the original trajectory, where the variable $k = k[n]$ indicates the non-uniformity of the sampling due to the dynamics of the trajectory causing similar distances to be traversed in quite disparate time intervals.

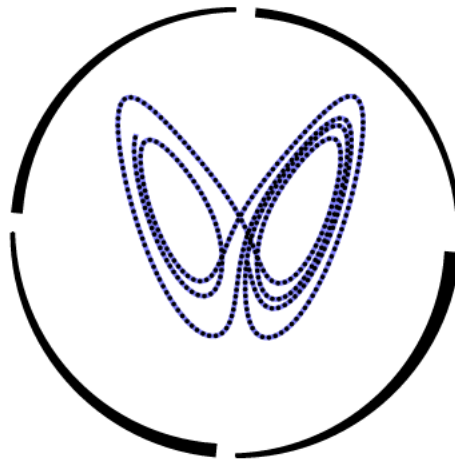


Figure 4.4: Sampling of the 2D projected trajectory by the NLCode Generator. At this stage of the NLCode’s creation there is no need for the colouring of the pattern. The trajectory and frame/plot specifications are the same as in Fig. 4.2, and the newly introduced parameter is the minimum interpoint distance D defined in Subsec. 4.2.3 by Eq. 4.9, with distance factor $f_D = 0.419$.

Figure 4.4 shows the sampling of the same trajectory shown in Fig. 4.2, for a specific value of the *minimum interpoint distance*, or simply *sampling distance* D given in the caption, in *plot coordinates*. D and a few other parameters that specify the processes of the Generator relating to SOs will be discussed in Subsec. 4.2.3.

4.2.3 General Method: Squares & Occupancies

The first method to take the sampled 2D trajectory as input and process it further is called *Squares and Occupancies (S&Os)*. Its name comes from the main element all tasks performed by it rely on, and a characteristic property of that element. The main element is a set of *square areas* – usually referred to simply as *squares* – defined around each sample point with their sides normal to the axes of the 2D

plot coordinate system – and *not* to the edges of the line representing the pattern. The characteristic property of these squares is the *total number of sample points* that occupy each of them, including their centers, called *square occupancy* and denoted by O_S .

The S&Os method is built to detect SOs as instances where one – trivially, or more segments run through the area covered by a square, such that at least one sample point of each of those segments occupies the same square, that is, the O_S of that square is at least equal to the number of the segments involved. That number, i.e. the *number of overlapping segments* causing a SO, is called the *multiplicity* M_O of a SO, such that $M_O = 1$ means that there is no SO present, $M_O = 2$ indicates the presence of a SO caused by two overlapping segments and so on. As already suggested, the multiplicity of a SO is generally at most equal to the occupancy O_S of the square detecting the overlap, that is, $M_O \leq O_S$, and the equality only holds true when the sample points that contribute to O_S – other than its center – *definitively and non-redundantly* indicate the presence of an overlap. The distinction between this latter class of points and the sample points generally occupying a square will be revisited near the end of this section (Fig. 4.9), as it is a vital attribute of the S&Os method. Since by design, S&Os detects SOs a certain way, this method will be considered successful only if it manages to detect *all* the SOs present in the 2D projected trajectory, as those were defined in the beginning of Sec. 4.2. The method's success is therefore dependent on its parametric configuration which is the main topic of the present section.

Figure 4.5 demonstrates the general idea behind S&Os using the same trajectory shown previously (Figs. 4.2 and 4.4), but for a much larger number of iterations, $N = 12,000$, in order to dramatically increase the number of SOs and fill up all *multiplicity slots* available (see legend of Fig. 4.5).

The set of square areas defined on the sampled trajectory is specified by two parameters, both expressed in plot coordinates. The first one is the size of the squares, which is represented by their side S_{side} , and the second is the minimum interpoint distance D of the sample introduced in the previous subsection. The relative values of S_{side} and D play a key role in the setup of S&Os, but in order to properly weigh in on the process, at least one of them must be linked to some known spatial characteristic of the pattern. This link is established according to a few considerations which are presented below and illustrated in Figs. 4.6 – 4.8.

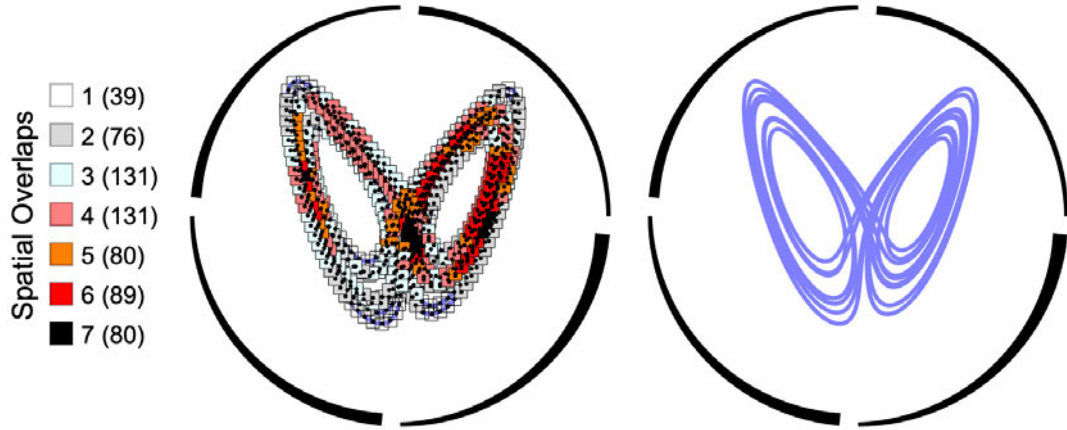


Figure 4.5: Visual demonstration of the Squares and Occupancies process via a worst case scenario of SOs. The trajectory is the same as in Figs. 4.2 and 4.4, but for a larger number of iterations, $N = 12,000$. The squares plotted around the sample points in the plot on the left are colour-coded based on the multiplicities of the SOs detected. As indicated, the use of a much larger N caused a dramatic increase of SOs of both types (self-crossings and contiguities). The colour scheme used tracks SOs with multiplicity 1 up to 6, and the 7th multiplicity slot (black), includes all occurrences of SOs with $M_O \geq 7$. The numbers in parentheses in the legend of the figure report the number of SOs in each multiplicity slot. The plot on the right is included for visual confirmation. Apart from the total number of iterations, the trajectory and frame/plot specifications are the same as in Figs. 4.2 and 4.4. The newly introduced parameters are the sampling distance D and the square size, which is represented by their side S_{side} , and are defined by Eq. 4.9 and Eq. 4.8 respectively, with distance factor $f_D = 0.628$ and square factor $f_S = 1.5$.

The fact that the general purpose of S&Os is to go through a series of 2D projected trajectories and accept or reject them using criteria based on SOs, makes the notion of a *smallest resolvable feature* – a term borrowed from the world of *optics* and *optical systems* – immediately relevant to this process. The smallest features that need to be visually resolved in the current implementation are line segments, which also happen to have a natural measure associated with them, that is, the line width L_{Plot}^{TH} . Specifically, assuming that two line segments run parallel to one another (contiguity), since intersecting segments (self-crossing) have no hope of being (spatially) resolved in the vicinity of the intersection, and taking into account that the sample points are located at the midpoint of the line width, the shortest distance between two segments that still allows them to be distinguished as two separate entities is naturally defined as the line width L_{Plot}^{TH} . This point is illustrated in Fig. 4.6a.

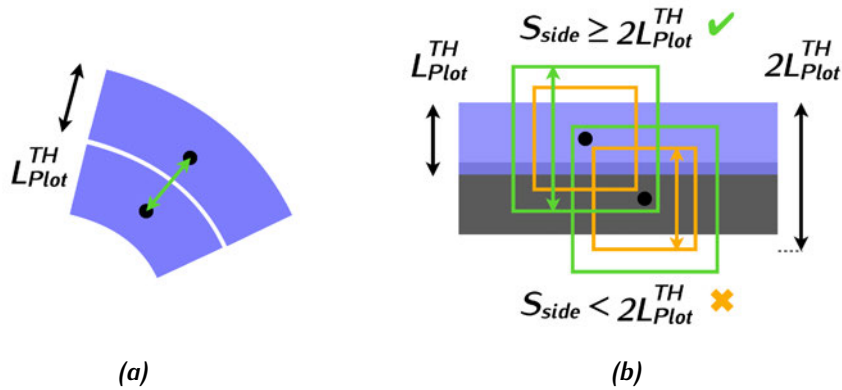


Figure 4.6: Line width Vs square size in the S&Os method. (a) The shortest distance between two contiguous line segments that still allows them to be distinguished as two separate entities is the line width L_{Plot}^{TH} . (b) The smallest size of the squares – represented by their side S_{side} – S&Os should implement is $2L_{Plot}^{TH}$, since it allows the presence (absence) of a sample point within a detecting square to determine whether a SO is present (absent) in the area covered by that square. This figure also shows a contrary case, where an existing overlap is missed because the sample point that should signify its presence falls outside a detecting square of less than the required size.

Each square is defined around a point of the sampled trajectory and its purpose is to enable the identification and counting of *other* sample points occupying it, as a means to detect SOs. In order to ensure that the S&Os method does not miss existing SOs due to it failing to cover a sufficient area around the segments tested, the squares used for that purpose should be no less than a certain size. Using the point made above, i.e. that the shortest distance between two line segments that allows them to be visually resolved is L_{Plot}^{TH} , Fig. 4.6b shows that the smallest size the squares should be allowed to have is $2L_{Plot}^{TH}$. This condition lets the presence (absence) of a sample point within a detecting square be an indication of the presence (absence) of a SO. On the contrary, if $S_{side} < 2L_{Plot}^{TH}$, the sample point that should indicate the presence of a quite substantial overlap would lie outside the test square and therefore be missed. The condition S_{side} is subject to, is mathematically expressed as

$$S_{side} \geq 2L_{Plot}^{TH}, \quad (4.2)$$

where L_{Plot}^{TH} was defined in Sec. 3.1 as $L_{Plot}^{TH} = \ell_{TH} S_{Plot}^{NLC}$ (Eq. 3.8), with S_{Plot}^{NLC} given by Eq. 3.9 (or Eq. 3.10), which simply involves other known parameters.

Having defined the minimum size S_{side} of the squares in terms of the line width L_{Plot}^{TH} (Eq. 4.2), allows the sampling distance D to be determined in relation

to either of these two quantities. Generally speaking, D can be as large (small) as one wishes it to be – within the bounds of any given trajectory (respecting the limits imposed by the finite iteration step Δt of the RK4 integrator), but for the purposes of S&Os the question is this: “Given L_{Plot}^{TH} and S_{side} , how large can D become before S&Os starts to fail detecting SOs?”. This immediately implies that D should also be defined in terms of a limiting value, but contrary to S_{side} , the sampling distance will have a maximum value D_{max} .

A geometrical formulation of the above question in search of D_{max} is schematically represented in Fig. 4.7, where it is seen to utilise a pair of overlapping segments, one of which (black) is used as a *reference* because the detecting squares are defined around its sample points, and the other (blue) as the *test* segment, since it contains sample points which must fall within the detecting squares in order to indicate the presence of a SO. It is assumed that:

1. The angle between the horizontal axis of the plot coordinate system and the reference segment is *any* θ_{ref} (counterclockwise), while the respective angle for the test segment is *any* θ_{test} .
2. The point of intersection – if there is such a point, since the lines treated here are of finite width and segments running parallel to one another ($\theta_{test} = \theta_{ref}$) can have zero up to infinite points of “intersection” – is located at the middle of the line segment connecting two reference sample points.
3. The square size takes its minimum value, i.e. $S_{side} = 2L_{Plot}^{TH}$.
4. The geometrical setup of the problem is “marginal”, i.e. it is such that the SO is borderline missed – if the distance between the sample points is increased even slightly.

In Fig. 4.7, as well as in the following discussion, the two types of SOs are addressed separately. Despite this fact, however, it will soon become clear that the search of D_{max} presented is by no means exhaustive. In the case of self-crossings (Fig. 4.7a), the “marginal” setup of the problem requires that the *longest* interval x defined by the intersection of the test segment’s bisector and the two squares be greater than or equal to the sampling distance D , that is, $x \geq D$. After close inspection of Fig. 4.7a, it can be shown that the above condition is equivalent to

$$D \leq \frac{2L_{Plot}^{TH}}{\sin \theta_{ref} + \sin \theta_{test}} = D_{max}(\theta_{ref}, \theta_{test}) \quad (\text{self-crossings}), \quad (4.3)$$

which defines D_{max} as a function of the two segments’ angles θ_{ref} and θ_{test} .

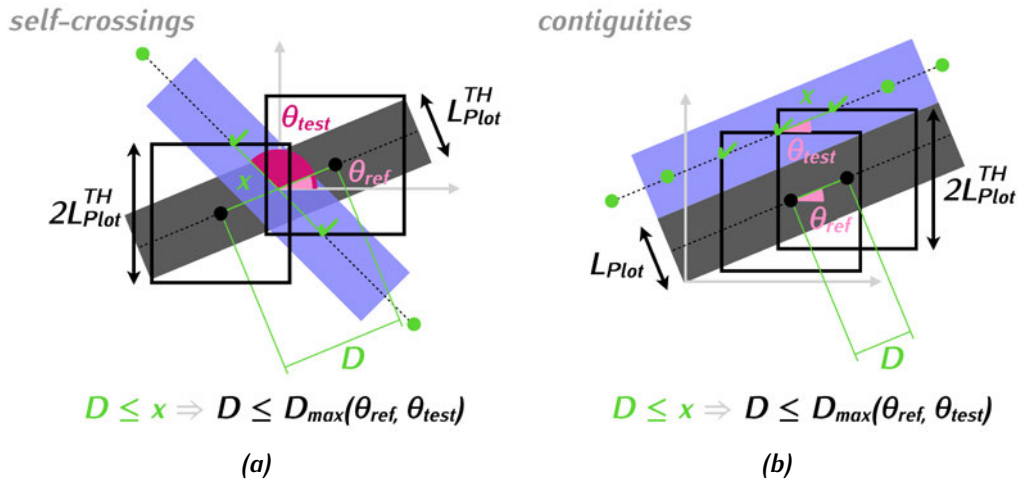


Figure 4.7: Partial derivation of the maximum sampling distance D_{max} as a function of the angles θ_{ref} and θ_{test} formed by the horizontal axis of the plot coordinate system and the reference (black) and test (blue) segments respectively. The figures present the “marginal” geometrical setup used in the derivation of $D_{max} = D_{max}(\theta_{ref}, \theta_{test})$ for the two types of SOs. The term “marginal” setup implies that S&Os will fail to detect the overlap present if the sampling distance D becomes even slightly greater than x , which is defined differently for self-crossings and contiguities. The detecting squares belong to the reference sample points (black) and the test sample points (green) that signify the detection of the SOs are shown as ticks. (a) Self-crossings: The condition $x \geq D$ with x the longest interval defined by the test segment’s bisector and the two reference squares, leads to the definition of D_{max} given by Eq. 4.3. (b) Contiguities: In this case $\theta_{test} = \theta_{ref}$. The condition $x \geq D$ with x in this case the shortest respective interval, leads to a different expression for D_{max} , given by Eq. 4.4. This derivation is incomplete (partial), since the two cases examined are sufficient to reveal that D_{max} is a piecewise function of the two angles, but fail to define it over its entire domain $[0, 2\pi] \times [0, 2\pi]$.

In the case of contiguities the situation differs slightly. According to Fig. 4.7b, the “marginal” setup of the problem requires that it is the *shortest* interval x defined by the intersection of the test segment’s bisector and the two squares that should be greater than or equal to the sampling distance D . In other words, the condition $x \geq D$ stays the same, but x is defined differently. Closer – in this case – inspection of Fig. 4.7b leads to a different condition for D and consequently, a different expression for D_{max} , i.e.

$$D \leq \left(1 - \tan \frac{\theta_{ref}}{2}\right) \frac{L_{Plot}^{TH}}{\cos \theta_{ref}} = D_{max}(\theta_{ref}, \theta_{test}) \quad (\text{contiguities}), \quad (4.4)$$

where the two arguments of D_{max} remain in order to maintain a unified approach despite the special cases treated.

The results just presented invite a few important remarks:

- Defining D_{max} as a function of θ_{ref} and θ_{test} means that it is a *variable* quantity. For the sampling process of the 2D projected trajectory presented in Subsec. 4.2.2, this means is that D_{max} would have to be calculated multiple times for each sample point *while sampling the trajectory*.²⁸ Alternatively, if $D_{max}(\theta_{ref}, \theta_{test})$ has a *global minimum* value $(D_{max})_{min}^{glob}$, a perfectly viable option would be to set D to some value less than or equal to that global minimum, for which S&Os cannot fail under any circumstances. Maybe counterintuitively but nonetheless in essence, this option makes the search of a *constant* D_{max} a *minimisation problem*, which brings forth the next important remark.
- Having derived *two* functional dependencies of D_{max} on θ_{ref} and θ_{test} means that D_{max} is a *piecewise function*; the seemingly inconsequential distinction between the two expressions given by Eqs. 4.3 and 4.4 using the two types of SOs, namely *self-crossings* and *contiguities*, silently defines two different *domains of applicability* for those expressions, with respect to the two angles θ_{ref} and θ_{test} . If at all present, the minima of D_{max} as a piecewise function are bound to be *local*, denoted by $(D_{max})_{min}^{loc}$, which explains the earlier mention of a global minimum as the proper assignment for D . Another earlier mention, however, concerning the fact that the cases presented in Fig. 4.7 do not represent an exhaustive search of D_{max} , implies that the two expressions derived above – with their implied sub-domains – do not cover the entire domain $[0, 2\pi] \times [0, 2\pi]$ this piecewise function is expected to have. In other words, the above results do not include all the pieces of D_{max} , and θ_{ref} and θ_{test} are not *any* angles as stated in the initial assumptions.

If the purpose of deriving the complete piecewise function $D_{max} = D_{max}(\theta_{ref}, \theta_{test})$ is to gain access to its local minima and from those obtain its global minimum $(D_{max})_{min}^{glob}$ allowing D to be defined as a constant, as is the case here, it is worth noting that (a) it takes a laborious effort to derive all the pieces of $D_{max} = D_{max}(\theta_{ref}, \theta_{test})$, which is “*quadrupled*” by the fact that the subsequent calculation of its local minima requires the employment of the actual *definition*

²⁸What one would do in this scenario is start with the first sample point – used as reference – and then find the next one through trial and error, i.e. by testing other points until one finds the farthest point for which the condition $D \leq D_{max}(\theta_{ref}, \theta_{test})$ is met. This process, which includes the calculation of the two angles for every test point, would have to be repeated for every point in the sampled trajectory.

of one-sided limits of multivariate functions – instead e.g. of setting two partial derivatives to zero and solving for the two angles, and (b) the cases examined geometrically during the derivation of $D_{max} = D_{max}(\theta_{ref}, \theta_{test})$ inevitably *reveal* its actual minima before the function itself even gets written down. Adding to that the fact that the results produced by S&Os will ultimately be subjected to trajectory acceptance/rejection criteria of a statistical nature (Subsec. 4.2.4), renders such a quest futile, at best.

The two local minima competing for the place of D_{max} 's global minimum are presented in two critical cases of SOs, one of which is a self-crossing and the other a contiguity. Figure 4.8 schematically represents these two cases and derives the respective conditions D must fulfill for each SO to be borderline missed. Specifically, the self-crossing shown in Fig. 4.8a has the reference segment at 45° with the horizontal axis of the plot coordinate system, and the test segment is normal to the reference, i.e. $\theta_{ref} = \pi/4$ and $\theta_{test} = 3\pi/4$. Similarly to the preceding discussion, requiring that $x \geq D$ leads to a condition that defines the first local minimum of D_{max} , that is,

$$\text{Self-crossing at } (\theta_{ref}, \theta_{test}) = (\pi/4, 3\pi/4) : D \leq \sqrt{2} L_{Plot}^{TH} = (D_{max})_{min}^{loc, SC-45} \quad (4.5)$$

In the case of the contiguity shown in Fig. 4.8b, both segments are at 45° with the horizontal, that is, $\theta_{ref} = \theta_{test} = \pi/4$ and the condition $x \geq D$ leads to the second local minimum of D_{max} , i.e.

$$\text{Contiguity at } (\theta_{ref}, \theta_{test}) = (\pi/4, \pi/4) : D \leq \frac{2}{1 + \sqrt{2}} L_{Plot}^{TH} = (D_{max})_{min}^{loc, C-45} \quad (4.6)$$

The two local minima given in Eqs. 4.5 and 4.6 are differentiated by the additional superscripts *SC-45* and *C-45*, which stand for *Self-Crossing* and *Contiguity* at 45° respectively. The global minimum of D_{max} is the smallest of the two local minima found, i.e. $(D_{max})_{min}^{loc, C-45}$, and is assigned to the *maximum constant value* the sampling distance D is allowed to take to ensure that S&Os will not fail under any circumstances. D is therefore subject to the following condition:

$$D \leq (D_{max})_{min}^{glob} = \frac{2}{1 + \sqrt{2}} L_{Plot}^{TH}. \quad (4.7)$$

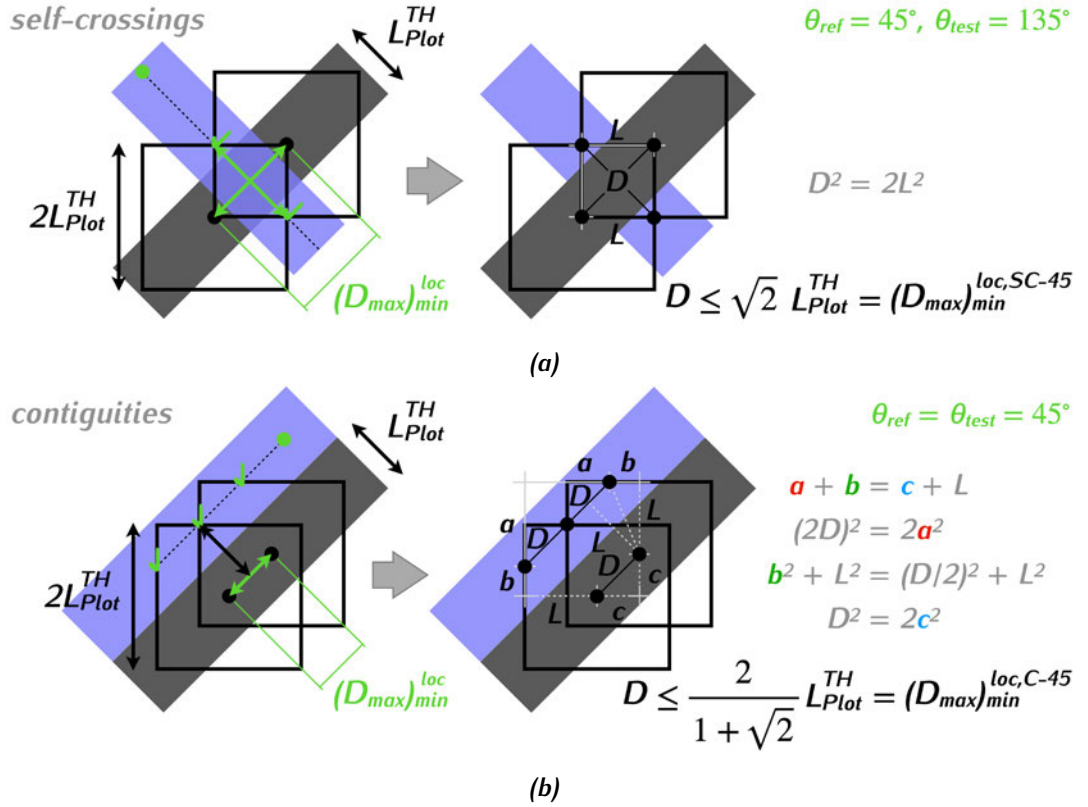


Figure 4.8: Calculation of D_{max} 's global minimum $(D_{max})_{min}^{glob}$ based on two critical cases of SOs. The self-crossing and contiguity presented are not simply "marginal" geometrical setups each corresponding to a piece of D_{max} as the piecewise function of θ_{ref} and θ_{test} it was shown to be (see main text and Fig. 4.7); the values given to the two angles are such that D_{max} is locally minimised, i.e. takes the respective (local) minimum values of two of its pieces. The figures on the left show the "marginal" setups following the same rules as Fig. 4.7, and the figures on the right present an overview of the calculation of D_{max} 's local minima in terms of the line width L_{Plot}^{TH} , both schematically and symbolically. (a) Self-crossing with $(\theta_{ref}, \theta_{test}) = (\pi/4, 3\pi/4)$: The local minimum is denoted by $(D_{max})_{min}^{loc, SC-45}$ and is defined as shown in the figure on the right (also given by Eq. 4.5). (b) Contiguity with $(\theta_{ref}, \theta_{test}) = (\pi/4, \pi/4)$: The local minimum is denoted by $(D_{max})_{min}^{loc, C-45}$ and is defined in the respective figure (also see Eq. 4.6). The global minimum of D_{max} is the smallest of the two local minima, that is, $(D_{max})_{min}^{glob} = (D_{max})_{min}^{loc, C-45} \simeq 0.83 L_{Plot}^{TH}$. By not allowing the sampling distance D to become greater than $(D_{max})_{min}^{glob}$, two important tasks are achieved: First and foremost, it is ensured that S&Os will not fail at detecting SOs under any circumstances and second, having a constant maximum allowed D at its disposal, the sampling of the 2D projected trajectory is allowed to remain the simple process described in Subsec. 4.2.2.

Based on the conditions S_{side} and D are subject to, given by Eqs. 4.2 and 4.7 respectively, these parameters are defined in terms of L_{Plot}^{TH} as

$$S_{side} = f_S (2L_{Plot}^{TH}) , \quad (4.8)$$

and

$$D = f_D \left(\frac{2}{1 + \sqrt{2}} L_{Plot}^{TH} \right) , \quad (4.9)$$

where $f_S \geq 1$ and $f_D \leq 1$ are multiplicative factors called *square factor* and *distance factor* respectively.

In the beginning of this section, the inequality $M_O \leq O_S$ between the multiplicity of a SO and the occupancy of the square detecting the overlap was attributed to a distinction between two classes of sample points. The first class consists of M_O sets of points, each of which corresponds to a segment running through the detecting square. The sample points in those sets (a) form sequences on the sampled trajectory, (b) belong to the segments causing the SO, and (c) fall within the detecting square. The points forming this class are collectively referred to as *sequential points*, and their total number equals the occupancy O_S of the detecting square. The second class of points is formed by letting *one* sample point from each of the above sets represent the segment it belongs to. The total number of those points is equal to the multiplicity M_O of the SO, and since they *definitively and non-redundantly* indicate its presence on the 2D projected trajectory, they are called *overlap points*.

Figure 4.9 assists in making the distinction between sequential and overlap points by schematically representing three overlapping segments of a trajectory that was sampled according to the condition Eq. 4.7 imposes on the sampling distance D . While every sample point shown has a square defined around it, the figure only depicts three squares for clarity. The multiplicity of the SO detected by each square as well as its occupancy are also shown in labels.

The number of sequential points in any given square contributed by each of the segments causing a SO will vary, but generally depend on the density of the sample – controlled by the sampling distance D , *relative* to the size of the squares – controlled by S_{side} . By detecting, grouping and counting points, the main task of S&Os is to obtain the multiplicity M_O of each SO, i.e. the number of segments running through each square. Fortunately, obtaining M_O is as simple

a computational task as *indexing* the points of the sampled trajectory, in order to later group the total number of points occupying each square into points that form sequences; for each square, the number of those groups (sets) is the multiplicity M_O of the detected SO.

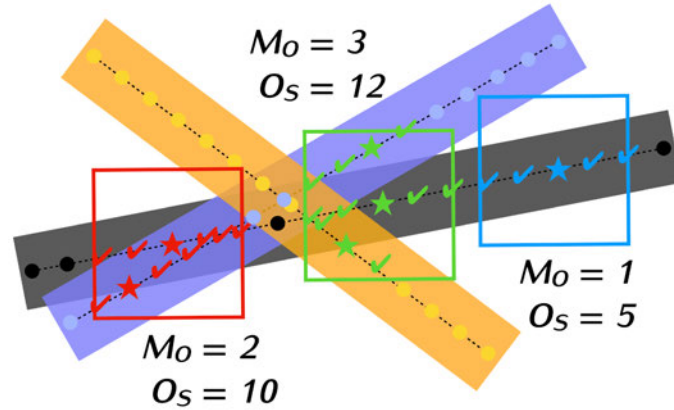


Figure 4.9: A detailed view of the inner workings of S&Os. The figure schematically represents the SOs caused by three overlapping segments using one reference (black) and two test segments (blue and yellow). The three detecting squares shown in red, green and blue are defined around three reference points (black – if not engaged). For clarity, the figure omits the squares centered around every single point depicted. The sampling distance D is less than the maximum allowed (Eq. 4.7), which causes a redundancy of sample points within each of the detecting squares. The sequential points match the colour of the square they fall into and are shown as ticks and stars, as opposed to points that do not belong to any of the three squares which are shown as bullet points. The overlap points – a subset of sequential points – are shown as stars and they include the squares' centers. As indicated by the labels of the squares, the occupancies O_S of the red, green and blue squares are 10, 12 and 5 respectively, reflecting the total number of sequential points each of them contains. The multiplicities of the SOs detected are 2, 3 and 1 respectively, and they reflect the number of overlap points in each square.

Figure 4.9 also highlights a different aspect of the important effect the relative values of D and S_{side} have on S&Os. Conditioning this method not to miss existing SOs is, admittedly, a “just cause”, but with a sampling distance as small as Eq. 4.7 dictates – always relative to the size of the squares, S&Os is bound to detect and record the same overlap multiple times. This shortcoming is remedied by deleting duplicate records of SOs, i.e. instances of squares containing the same sample points. Note that S&Os only reports based on unique records of squares, which means that the numbers in parentheses in the legend of Fig. 4.5 correspond – as stated – to SOs and not squares.

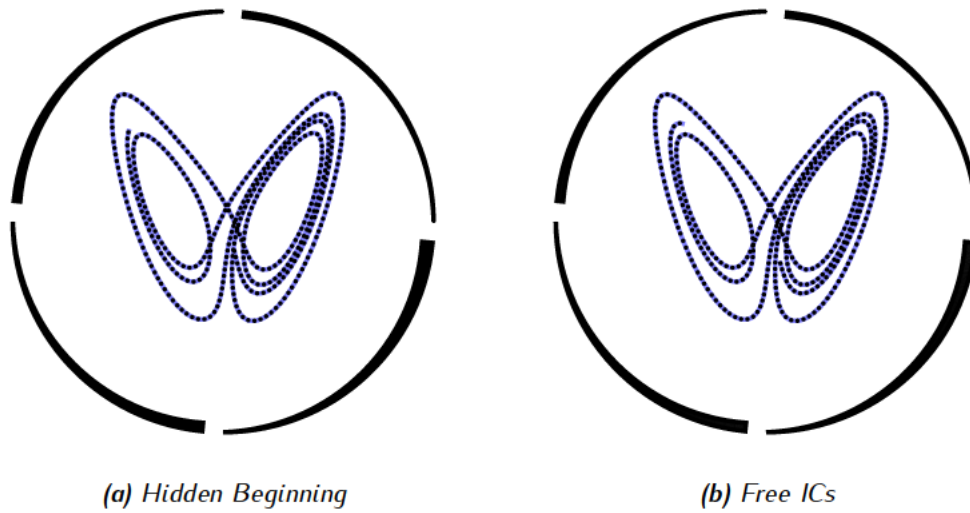


Figure 4.10: S&Os addresses the issue described in Fig. 4.2 as Hidden Beginning. The segment around the ICs of the sampled trajectory in Fig. 4.4, also shown as figure (a) here for comparison, overlaps with other segments of the trajectory. In figure (b) the trajectory begins at the first sample point that occupies its square on its own, and is therefore “free” from SOs. 3D ICs: $(x_0^1, x_0^2, x_0^3) = (-0.0381, -0.0846, -0.4230)$. As a result of the process recreating the trajectory with the same number of iterations, $N = 5,000$, and altering only the ICs, the treated trajectory has the same length as the original one.

After detecting the SOs present in the trajectory examined and before subjecting it to the acceptance/rejection criteria discussed in Subsec. 4.2.4, S&Os addresses the issue described in Fig. 4.2 as *Hidden Beginning*. This simple task consists of checking whether the beginning of the pattern, i.e. the segment around the point representing the trajectory’s ICs, is hidden by other segments of the trajectory due to SOs, and if the ICs are indeed hidden, finding the first point in the sampled trajectory that occupies its square on its own. The method then *resets* the ICs to the point found, and *starts over*, i.e. solves the system using the new ICs, samples the 2D projected trajectory and recalculates the multiplicities of the SOs present. This is done in an attempt to maintain as much of the total length of the pattern as possible; it is possible for the first “free” sample point to be so far down the trajectory, that S&Os ends up omitting a great part of it. This process is illustrated in Fig. 4.10.

4.2.4 Acceptance/Rejection Criteria

The data obtained from S&Os' search for SOs on a 2D projected trajectory come in the form of 7 integers, denoted by $N_{SO}^{M_O}$, with M_O indicating the multiplicity of a SO. Each of those numbers represents the number of SOs of multiplicity M_O found, after making sure the ICs are free of overlaps and deleting duplicate records. For $M_O = 1, \dots, 6$, and $(\geq)7$ the number of multiplicity slots are 7, and $N_{SO}^{M_O}$ are the numbers in parentheses in the legend of Fig. 4.5. The slot with $M_O \geq 7$ groups together all SOs with multiplicities 7 and above.

The ultimate purpose of S&Os is to determine the viability of a trajectory as an NLCode pattern, in terms of its spatial characteristics, i.e. SOs. Using the data obtained from its search, the method does this by applying two simple criteria. Let N_{SO} be the total number of SOs found, that is

$$N_{SO} = \sum_{M_O=1}^{(\geq)7} N_{SO}^{M_O}. \quad (4.10)$$

The fraction of SOs with multiplicity M_O is the ratio

$$r^{M_O} = \frac{N_{SO}^{M_O}}{N_{SO}}, \quad M_O = 1, \dots, 6 \text{ and } (\geq)7. \quad (4.11)$$

Considering that $M_O = 1$ indicates the absence of a SO, r^1 should ideally be equal to 1 – for a trajectory completely free of SOs. *Realistically*, however, for an acceptable trajectory r^1 is expected to be above some critical value r_{crit}^1 less than, but not too far from unit. On the contrary, since all $M_O > 1$ indicate the presence of SOs, in these cases r^{M_O} should *ideally* be 0, and *realistically* below some critical value $r_{crit}^{M_O}$ greater than, but not too far from zero. Tests performed on a series of candidate trajectories confirmed that in order to determine whether a 2D projected trajectory is too overwhelmed by SOs to be worth processing further, it suffices for it to be tested against the above criteria *only* for $M_O = 1$ and 2, i.e.

$$r^1 > r_{crit}^1 \quad (4.12a)$$

$$r^2 < r_{crit}^2. \quad (4.12b)$$

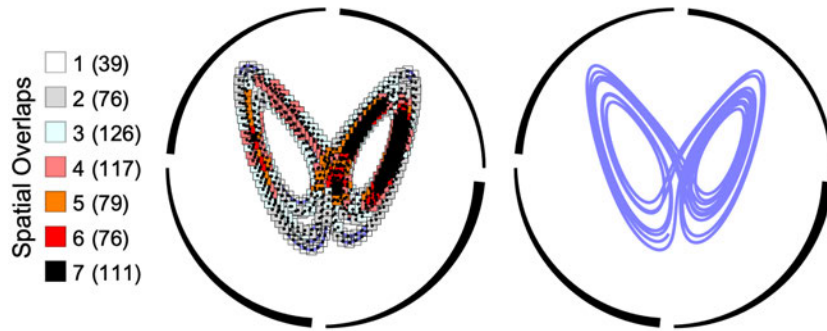
A set of values for r_{crit}^1 and r_{crit}^2 that seem to accept trajectories which are later confirmed to be viable NLCode patterns, are the following:

$$r_{crit}^1 = 0.8, \quad r_{crit}^2 = 0.05. \quad (4.13)$$

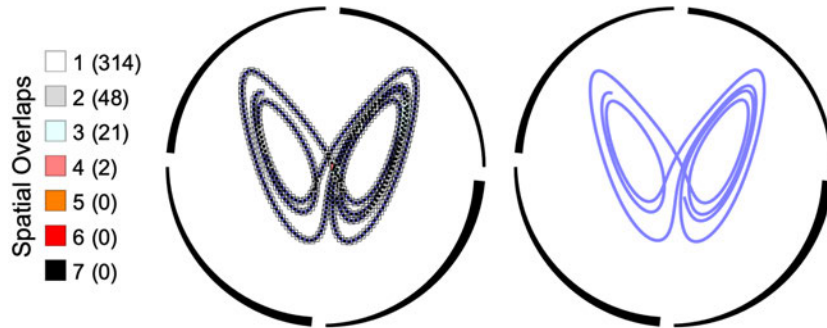
Figure 4.11 shows the above criteria in action not by listing entirely different trajectories that either pass or fail them, but by using a single trajectory as reference, start it from a state of complete rejection, and observe the drastic changes required in order to make it pass the first and then both criteria given by Eqs. 4.12 and 4.13. The trajectory used as reference is the same trajectory used so far (Figs. 4.2 – 4.5 and Fig. 4.10), and is shown in Fig. 4.11a in a state of complete rejection. This trajectory only differs from the one shown in Fig. 4.5 in that S&Os has treated the former for overlapping ICs, according to the process described at the end of Subsec. 4.2.3 (Fig. 4.10).

The initial trajectory is made to pass the first criterion by reducing the number of iterations from $N = 12,000$ to 5,000 and more importantly, the square factor from $f_S = 1.5$ to 1. Note that either one of these alterations alone would not suffice to make the trajectory pass this criterion. While reducing the number of iterations might be an acceptable remedy in many situations where the trajectory can afford it, changing the square factor can only be done during process standardisation, i.e. when the optimal values of the control parameters are still being assessed. This also brings forth the rather obvious observation that the real objective is not to make a trajectory pass the criteria by any means necessary, but to make it readable; contrary to what was done in this example, this may actually require to tighten the conditions, i.e. increase the square factor f_S instead of decreasing it.

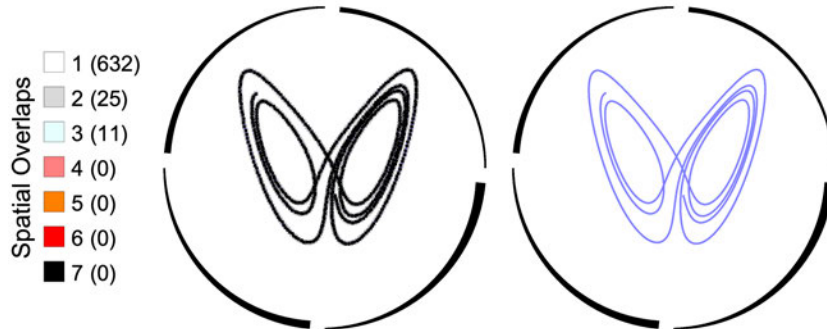
Having observed the effects of N and f_S on its performance, the trajectory was next made to pass the second criterion as well, by decreasing the thickness factor ℓ_{TH} from 0.01 to 0.006. As Eq. 4.8 suggests, this decrease has an effect similar to that of the square factor's decrease, and having already decreased f_S in the previous step, simply exacerbates it. The advantage of altering ℓ_{TH} instead of f_S is that, as a spatial characteristic of the pattern itself, ℓ_{TH} is more accessible than f_S which, as a parameter internal to the Generator, should be kept fixed across several tests performed on different trajectories. The range of allowed values for ℓ_{TH} is discussed in Sec. 4.4, where it is seen to span approximately one order of magnitude between 10^{-3} and 10^{-2} (Eqs. 4.31 and 4.37).



(a) Rejection with $r^1 = 0.0625 < r_{crit}^1 = 0.8$ and $r^2 = 0.1218 > r_{crit}^2 = 0.05$.
ICs after corrections $(-0.0381, -0.0846, -0.4230)$, $N = 12,000$, $f_S = 1.5$, $\ell_{TH} = 0.01$.



(b) Passed 1st criterion with $r^1 = 0.8156 > r_{crit}^1$, but failed 2nd with $r^2 = 0.1248 > r_{crit}^2$.
ICs after corrections $(0.0674, 0.1054, -0.1969)$, $N = 5,000$, $f_S = 1$, $\ell_{TH} = 0.01$.



(c) Acceptance with $r^1 = 0.9461 > r_{crit}^1$ and $r^2 = 0.0374 < r_{crit}^2$.
ICs after corrections $(0.0614, 0.0993, -0.1732)$, $N = 5,000$, $f_S = 1$, $\ell_{TH} = 0.006$.

Figure 4.11: Spatial overlaps: From complete rejection to pattern acceptance: (a) The reference trajectory passes from complete rejection due to failing both criteria given by Eqs. 4.12 and 4.13 to (b) passing the first criterion (Eq. 4.12a), to (c) passing both the first and the second criterion (Eq. 4.12b). The trajectory specifications – prior to the correction for overlapping initial conditions and unless otherwise stated in the captions – are the same as in Figs. 4.2, 4.4, 4.5 and 4.10.

4.3 Colour Overlaps – Input Palette

An NLCode pattern with no SOs at all would have all of its different sections sufficiently separated from one another, and this spatial separation would be perceived as a colour difference between the pattern and the white background of the feature. Unfortunately, an NLCode pattern cannot be entirely free of SOs. A colour difference between its overlapping segments, however, can be perceived the same way spatial separations are perceived, which means that colour similarities can be used in a way similar to the way SOs are used by S&Os. As soon as a trajectory passes the first round of tests concerning SOs, the next task of the NLCode Generator is to test it for colour overlaps.

A *Colour Overlap (CO)* is defined as a SO of segments with matching colours. Since the colouring of the pattern is the last fort against the undesired effect of SOs, the criterion here is strict: *There can be no COs*. In order for a rigid approach like this to be able to confirm trajectories as viable patterns of the NLCode, the trajectories that passed S&Os' tests need to be inspected for COs under several different colour schemes. According to [Sec. 3.4](#), the period P of the feature's colour profile is the appropriate parameter to vary in this case, since it controls the colour gradient formed on the pattern ([Fig. 3.12](#)).

The NLCode Generator begins the creation of the colour schemes that are to be tested by defining a set of periods as follows: Given an appropriately chosen interval $[P_{min}, P_{max}]$, which the periods of interest should be drawn from, and a desired number N_P of those periods,

$$P_i = \left\lfloor P_{min} + \underbrace{\frac{P_{max} - P_{min}}{N_P - 1}}_{\Delta P} (i - 1) \right\rfloor, \quad i = 1, \dots, N_P, \quad N_P > 1, \quad (4.14)$$

defines a sequence of periods $\{P_i\}$ in ascending order. The *floor function*, denoted by $\lfloor \cdot \rfloor$, rounds down its argument to the nearest integer²⁹, which means that the periods P_i are defined as integers after all (see end of [Sec. 3.4](#)) – a choice made simply for convenience. The minimum and maximum periods are defined with respect to the length of the trajectory, i.e. the number of iterations N , by equations

$$P_{min} = \frac{N}{n_{rep}^{max}}, \quad \text{and} \quad P_{max} = \frac{N}{n_{rep}^{min}}, \quad (4.15)$$

²⁹Formally, the floor function of a real number returns the greatest integer that is less than or equal to its argument.

where the divisors n_{rep}^{max} and n_{rep}^{min} indicate the maximum and minimum number of times the colour base \mathcal{B} (Eq. 3.63) of the profile is to be repeated on the pattern respectively. The suggested values for these parameters are $n_{rep}^{max} = 20$ and $n_{rep}^{min} = 3$, but they can be set to any reasonable values between N and 1 deemed appropriate, in any given application. Lastly, the number of periods tested should be relatively large, for example $N_p \simeq 10$, or even larger, in order to give trajectories a chance to pass the COs test.

A trajectory that passed the SOs test needs to be examined for COs under every colour scheme defined by the periods in the above sequence $\{P_i\}$. In order to do that, the Generator makes use of S&Os' results. For every colour scheme defined, a new process begins at the top level of S&Os' structured output, which is the multiplicities M_O of the detected SOs, and descends to the individual squares that have detected SOs with $M_O > 1$ – no point in looking for COs in the absence of SOs, before it can calculate and compare the colours of the *overlap points* (Subsec. 4.2.3) contained in each square.

The *calculation* of the gradient colour \mathbf{c}_n an overlap point has under a given colour scheme of period P_i , is done using the interpolation formula defined in Sec. 3.4 (Eq. 3.66). It is reminded that the subscript n marks the indexed position of the overlap point in the *complete* 2D projected trajectory, i.e. *before* the sampling.

Colour models represent colours as vectors, and in the case of the RGB colour model used in this study, these vectors are 3D and their three components correspond to the *red*, *green* and *blue channels*. Generally speaking, colour *comparison* can be performed simply by using the *Euclidean distance* $d(\mathbf{c}_i, \mathbf{c}_j)$ between colours $\mathbf{c}_i = (c_i^1, c_i^2, c_i^3)$ and $\mathbf{c}_j = (c_j^1, c_j^2, c_j^3)$, which is defined as

$$d(\mathbf{c}_i, \mathbf{c}_j) = \|\mathbf{c}_i - \mathbf{c}_j\| = \sqrt{(c_i^1 - c_j^1)^2 + (c_i^2 - c_j^2)^2 + (c_i^3 - c_j^3)^2}. \quad (4.16)$$

The colour profile of the NLCode, however, forms a smooth gradient (Sec. 3.4), which means that expecting $d(\mathbf{c}_i, \mathbf{c}_j)$ to be *exactly zero* in order to match colours \mathbf{c}_i and \mathbf{c}_j is not the best practice; a process that reports a mismatch between two colours based on the fact that $d(\mathbf{c}_i, \mathbf{c}_j) \neq 0$, even though $d(\mathbf{c}_i, \mathbf{c}_j)$ may be less than, say 10^{-4} , would utterly fail at detecting any COs at all. One could instead define a critical value d_{crit} close to zero and demand that $d(\mathbf{c}_i, \mathbf{c}_j) > d_{crit}$ before recording a colour mismatch, but determining an appropriate value of d_{crit} can also be troublesome.

A more robust way to compare colours is to define a *colour palette*, i.e. a short sequence of appropriately chosen distinct colours, and then assign to each of the overlap points tested, the palette colour closest to the *gradient* colour c_n of that point. In practice, this is done by calculating the distance d between the gradient colour c_n of an overlap point and every colour of the palette (Eq. 4.16), and identifying the palette colour that corresponds to the minimum of the calculated distances. The fact that each palette colour has a certain index in the palette sequence conveniently makes colour (mis)matching a simple matter of comparison between integers.

The colour palette is also referred to as *input palette*, complementing the *output palette* introduced in Par. 5.2.2.3. The input palette is denoted by \mathcal{P}_{in} , and an obvious choice for it is either one of the colour bases \mathcal{B}^{Hue} and \mathcal{B}^{TM} defined in Sec. 3.4 (Eqs. 3.63, 3.67 and 3.68), including *only the first N_{in} distinct colours in them*, i.e.

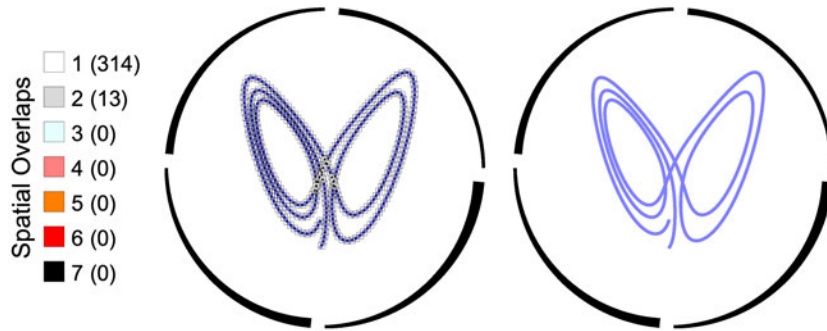
$$\mathcal{P}_{in} = \{\mathbf{b}_k\}, \text{ for } \mathbf{b}_k \in \mathcal{B} \text{ and } k = 1, \dots, N_{in}. \quad (4.17)$$

Once each overlap point has been associated with the index k of one of the colours in the palette sequence, a colour mismatch between two points is confirmed if

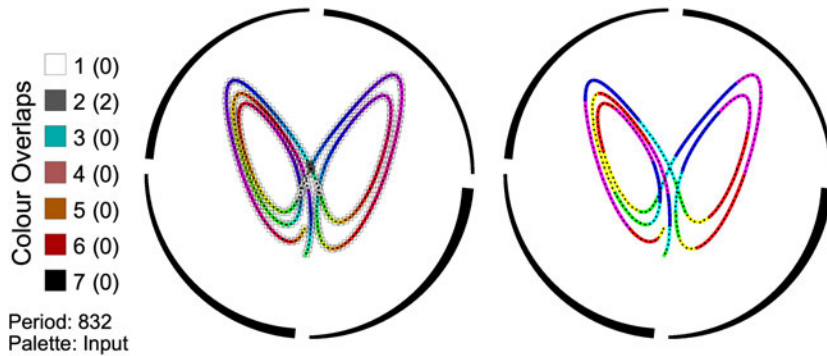
$$|k_i - k_j| \geq \kappa, \quad (4.18)$$

where $\kappa \geq 1$ is the minimum index distance two colours can have in order to be considered distinct or non-overlapping. The obvious choice in the case of \mathcal{P}_{in} is to set $\kappa = 1$, since both colour bases are relatively small. Generally speaking, the less contrasting – or more similar – the colours in a palette are, the more strict this condition can become, by setting κ to a higher value.

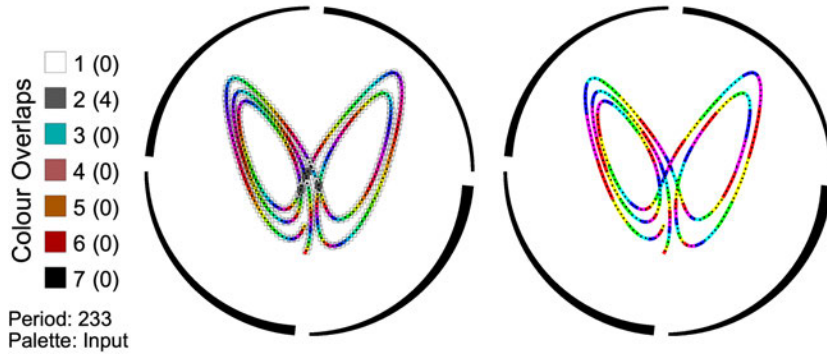
In summary, for every colour scheme defined by sequence $\{P_i\}$, the Generator accesses the multiplicity slots with $M_O > 1$ and at least one square in them – since the presence of a SO is a necessary condition for a CO to exist. It then descends to the square level of each multiplicity slot worth testing, and accesses the overlap points of each square in that slot. A pairwise comparison of the palette colours assigned to each of the overlap points occupying a square based on the condition given by Eq. 4.18, determines whether a CO is present or not. If at least one pair of overlap points is found to have matching colours, the entire colour scheme is discarded and the Generator moves on to the next period P_i . The two-page Fig. 4.12 demonstrates the results of this process with a few characteristic examples of colour schemes that either failed or passed the COs test.



(a) SOs: Acceptance with $r^1 = 0.9602 > r_{crit}^1 = 0.8$ and $r^2 = 0.0398 < r_{crit}^2 = 0.05$
 ICs after corrections $(-0.0308, -0.0675, -0.3341)$, $N = 4,000$, $f_S = 1$, $\ell_{TH} = 0.01$.

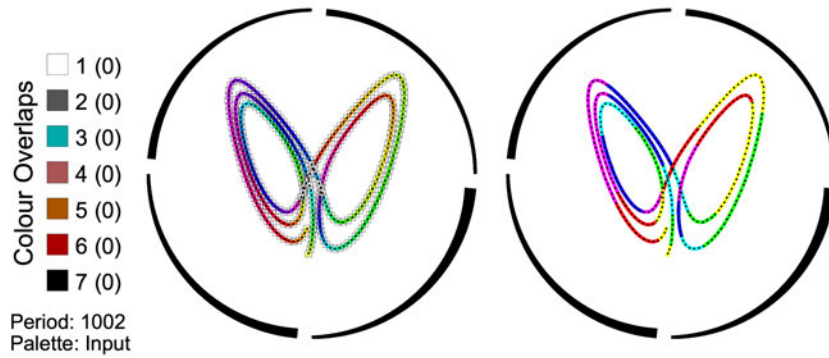


(b) COs: Rejection with 2 COs in slot with SO multiplicity of $M = 2$.
 Colour scheme: $P = 832$ – slow gradient, $\mathcal{P}_{in} = \mathcal{B}^{Hue}$ (Eq. 3.67).

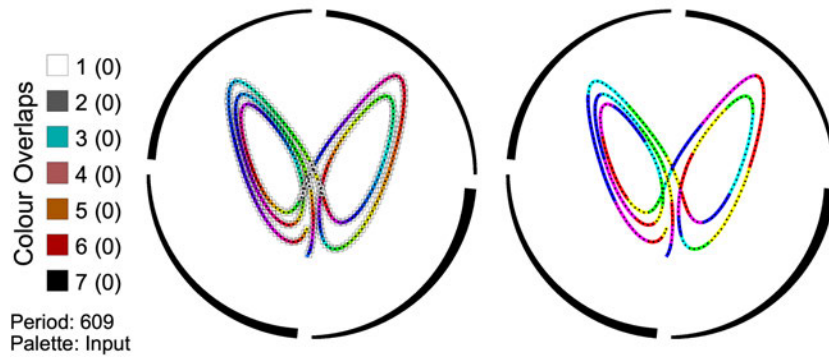


(c) COs: Rejection with 4 COs in slot with SO multiplicity of $M = 2$.
 Colour scheme: $P = 233$ – fast gradient, $\mathcal{P}_{in} = \mathcal{B}^{Hue}$.

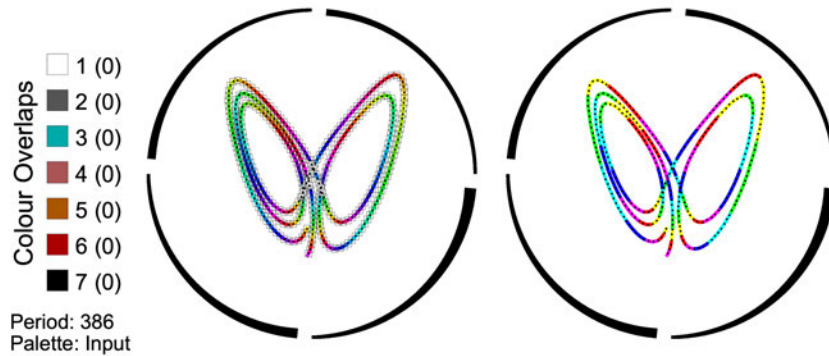
Figure 4.12: Colour overlaps: Varying the period P of the feature’s colour profile allows the creation of one or more colour gradients which render the pattern completely free of COs. Since spatial separations are perceived as colour differences between the pattern and its white background, relieving a pattern of COs provides a workaround to the undesired effect of SOs. (a) Trajectory passes the SOs test. (b) Colour scheme with large period is discarded due to COs. (c) Discarded colour scheme with small period.



(d) COs: Acceptance with 0 COs in all SO multiplicity slots.
Colour scheme: $P = 1002$ – slow gradient, $\mathcal{P}_{in} = \mathcal{B}^{Hue}$ (Eq. 3.67).



(e) COs: Acceptance with 0 COs in all SO multiplicity slots.
Colour scheme: $P = 609$ – moderate gradient, $\mathcal{P}_{in} = \mathcal{B}^{Hue}$.



(f) COs: Acceptance with 0 COs in all SO multiplicity slots.
Colour scheme: $P = 386$ – fast gradient, $\mathcal{P}_{in} = \mathcal{B}^{Hue}$.

Figure 4.12 (cont.): Colour overlaps: The colour-coded squares detecting SOs (Fig. 4.5) are overlaid with a darker theme to indicate the presence of COs, the number of which in each multiplicity slot is shown in parentheses. In the figures to the right, the gradient colours have each been replaced by the colour of the input palette closest to it (see main text). Figures (d), (e) and (f) show successful colour schemes with large, medium and small periods, which correspond to slow, intermediate and fast gradients respectively.

The last task of the Generator before rendering the final form of NLCode, is to address the aesthetic issue referred to as Abrupt Ending in [Sec. 4.1](#) ([Fig. 4.2](#)). According to the definition of the NLCode’s colour profile, the first colour in all patterns is *yellow*, that is, provided the ICs are free of SOs, which at this stage of the generation process has already been taken care of. An *Abrupt Ending* is defined as an instance where the pattern ends with any colour other than yellow, and it is addressed simply by finding the position of the last yellow point in the 2D projected trajectory, update N to that number and thereon only render the trajectory up to that point. Note that for a *slow to moderate gradient*, i.e. a large to medium period P (see e.g. [Fig. 4.12b](#), [Fig. 4.12d](#), and [Fig. 4.12e](#)), this process may have to discard large portions of the trajectories, and end up reducing the length of the patterns significantly. This is why *fast gradients* (see e.g. [Fig. 4.12f](#)) – but not too fast ([Fig. 4.12c](#)) – are generally preferred, and one of the reasons the Generator at the present stage of development offers a few choices regarding the final form of the NLCode. [Figure 4.13](#) shows the effect of this last process on a trajectory. The final result is a pattern that gradually emerges from the white background in the beginning and fades into it at the end, thus taking a small amount of effort to locate its two ends.

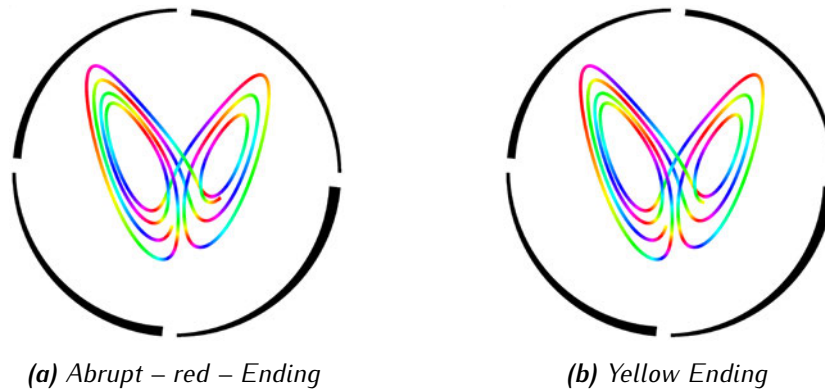


Figure 4.13: The Generator addresses the issue described in [Fig. 4.2](#) as Abrupt Ending. The first colour in all patterns is yellow; even if its initially hidden, this attribute becomes unequivocally visible after a pattern is treated for Hidden Beginning ([Fig. 4.10](#)). The last task of the Generator is to make sure the patterns also end in yellow. This is an aesthetic issue that once treated, it allows patterns to fade in and out of the background, sometimes making it seem like they are closed curves with no beginning and end. Both figures show the same trajectory earlier treated for COs ([Figs. 4.12](#) and [4.12f](#) in particular), but for $N = 5,468$. Figure (a) on the left shows the pattern before being treated for Abrupt Ending, where it is seen to end in red, and figure (b) on the right after the treatment.

4.4 Bridging Worlds: NLCode's Operating Window

The NLCode – *NLC* in notation – is a feature that is generated digitally with the purpose to be printed in a given *print size* S_{Print}^{NLC} , and be captured by a digital camera with certain specifications (Table 4.1), at some *working distance* W from the feature. In this context, the term *Operating Window (OW)* refers to:

- The range of allowed print sizes of the NLCode for a given working distance, *and* conversely,
- The range of allowed working distances for a given print size of the feature.

As the following analysis will show, the theoretical estimation of the NLCode's OW requires the employment of quite a few important concepts, models and formulas, most of which relate to the capabilities of digital photography to faithfully reproduce images of physical objects. However, before the NLCode becomes a *physical object*, it needs to be digitally created, which is why the three-step process *generation–printing–capture* outlined above starts with a *digital* object, which is transferred to the *analog* world via printing, and is then captured as a *digital* object once again.

Notice that Eq. 3.8 (Sec. 3.1) and Eq. 4.19 below, both express the same relationship between the line width and the feature's size, but each does so in terms of different "*instances*" of these quantities; the former is written with respect to the plot coordinate system and the latter with reference to the printing process (also see the [comment on notation](#) earlier made), i.e.

$$L_{Print}^{TH} = \ell_{TH} S_{Print}^{NLC}, \quad (4.19)$$

where L_{Print}^{TH} is the physical line width of the pattern when printed, and ℓ_{TH} the thickness factor introduced in Sec. 3.1. Equation 4.19 reveals the silent – yet reasonable – assumption that the proportionality constant between L_{Plot}^{TH} and S_{Plot}^{NLC} , i.e. the thickness factor ℓ_{TH} , is not affected by the conversion of the digitally generated NLCode to its printed version. This equation is important for the following reasons:

1. It is, in essence, a relationship innate to the first digital medium, i.e. the NLCode Generator, whose effect is transferred to the analog world.

2. As will soon become clear, L_{Print}^{TH} is a quantity directly related to *limits of resolution* in general and as such, is of highest importance to both printing and capture. Therefore, through L_{Print}^{TH} , Eq. 4.19 also provides a link between the analog and the second digital medium.
3. Last but not least, the thickness factor ℓ_{TH} is the only parameter the Generator heavily relies on, that has not yet been specified. By making it a part of the present analysis, Eq. 4.19 creates the opportunity for certain choices made with regard to ℓ_{TH} 's range of suitable values, to be properly addressed and substantiated.

For the above reasons, Eq. 4.19 is thought to form a “bridge” between the different media associated versions of the NLCode, and the best formula to begin a *parametric study* that involves all three of them.

Given a particular camera with a lens of focal length f and assuming a distance W between the captured object and the camera sensor, often called working distance, the ratio of the size of the object on the sensor S_{Sens}^{Obj} to the physical size of the object S_{Phys}^{Obj} equals the ratio of f to W . This relationship is mathematically expressed as

$$\frac{S_{Sens}^{Obj}}{S_{Phys}^{Obj}} = \frac{f}{W}, \quad (4.20)$$

and is derived based on the simplest *camera model*, called *pinhole camera* (Jähne, 2004). Note that the letter ‘S’ is used here to denote size, which can refer to either width or height, as long as this choice is consistent throughout all calculations.

The *resolution* of a digital camera is usually given as either the number of pixels in the horizontal, N_{Pixel}^H , and vertical, N_{Pixel}^V , dimensions of the sensor, e.g. ($N_{Pixel}^H \times N_{Pixel}^V$) pixels, or as the evaluated product N_{Pixel} of these two numbers in megapixels (MP), along with the sensor’s aspect ratio $a:b$, i.e. $10^{-6} N_{Pixel}$ MP, in $a:b$ ratio. The camera used for the development of the NLCode for example, is a 12 MP camera in 4:3 ratio, or 4032×3024 pixels (Table 4.1). It is an obvious fact that for a picture of a physical object to contain its entire image without *cropping* any parts of it, the object’s size on the sensor must not exceed the smallest of the two sensor dimensions, i.e.

$$S_{Sens}^{Obj} \leq \min [N_{Pixel}^H, N_{Pixel}^V] \cdot S_{Sens}^{Pixel}. \quad (4.21)$$

Notice that, since the above inequality refers to sizes on the sensor, it does not involve the distance W between the camera and the object, even though, for a

given size of the physical object, it is W which must be adjusted in order to satisfy the inequality.

Cropping is the first undesired effect of moving the camera too close to the object. If that is ignored, or if the object is too small to cause this issue, the next possible problem is the inability of the camera to focus on the object. The last camera specification of relevance to this study is the *minimum focusing distance* W_{min}^{Abs} which, for a camera equipped with a particular lens, is the minimum distance between the object and the camera that allows an optimal focus on the object. The obvious limit set by W_{min}^{Abs} on the working distance is expressed by the following inequality

$$W \geq W_{min}^{Abs}. \quad (4.22)$$

Table 4.1 lists the specifications of the camera used for the development of the NLCode that are necessary for the theoretical estimation of the NLCode's OW.

Table 4.1: Specifications of interest of the iPhone 6S Plus' primary (rear) camera (Device Specifications, *n.d.*).

| | |
|--|-----------------------------|
| Model | Sony IMX315 Exmor RS |
| Focal Length f | 4.02 mm |
| Pixel Size S_{Sens}^{Pixel} | 1.19×10^{-3} mm |
| Camera Resolution ($N_{Pixel}^H \times N_{Pixel}^V$) | (3024 \times 4032) pixels |
| or N_{Pixel} in $a:b$ | 12.19 MP in 4:3 |
| Minimum Focusing Distance W_{min}^{Abs} | 50 mm |

One of the first quantities of interest to any system relying on the processing of images captured by a digital camera is the *Smallest Resolvable Feature (SRF)* that can be utilised by that system. In Subsec. 4.2.3 it was argued that the smallest features of the NLCode that need to be visually resolved are *transverse* line segments, and the shortest distance between two segments that still allows them to be distinguished as two separate entities, i.e. the SRF, was defined as the line width L_{Plot}^{TH} of the pattern (Fig. 4.6a). However, Subsec. 4.2.3 makes a heuristic use of the term SRF, and the context of SOs and the S&Os process used to identify them is quite different from the present context, which this term properly belongs to. In order to be visually resolved, the line width L_{Print}^{TH} on a printed NLCode must be *at least* equal to the respective size of the SRF, i.e.

$$L_{Print}^{TH} \geq S_{Print}^{SRF}. \quad (4.23)$$

According to the Nyquist-Shannon sampling theorem (see e.g. Couch II, 2013), a *CT, bandlimited* signal can be perfectly reconstructed by samples taken at a rate f_s greater than twice the signal's highest frequency content B ³⁰, i.e.

$$f_s > 2B \xrightarrow{f_s=1/T_s} T_s < \frac{1}{2B}, \quad (4.24)$$

where f_s is called *sampling frequency* or *sampling rate* and is measured in *Hertz (Hz)*, and $T_s = 1/f_s$ is the *sampling interval*, measured in *seconds*. Equation 4.24 is often referred to as *Nyquist criterion*, the *minimum sampling rate* $f_{s,\min} = 2B$ *Nyquist rate* or *Nyquist frequency*, and similarly the *maximum sampling interval* $T_{s,\max} = 1/2B$ *Nyquist sampling interval*.

The sampling theorem is stated the same way regardless of whether it is applied to signals in 1D, such as audio or electrical signals, or in 2D, such as images (Gonzalez & Woods, 2008). This is despite the fact that the independent variable in the former type of signals is time, and in the latter is space in 2D. This suggests that the quantities *sampling rate* f_s and its conjugate *sampling interval* T_s , may be subject to different interpretations in the case of images. Taking as reference the right hand side version of Eq. 4.24 expressing the Nyquist criterion, note that T_s is the interval representing the “*shortest addressable element*” in the *discrete* or *discretised* (sampled) signal, while $1/B$ is the interval representing the “*shortest lived occurrence*” in the original, *CT* (unsampled) signal, as that is measured on the recording instrument. In the case of digital images, the “*smallest addressable element*” in the image is the *pixel*, and the “*smallest occurrence*” on the original scene captured is the *SRF*, as that is measured on the camera sensor. By letting the size S_{Sens}^{Pixel} of the pixels on any given sensor replace T_s in the above inequality, and the size S_{Sens}^{SRF} of the SRF on the sensor replace $1/B$, the Nyquist criterion reads

$$S_{Sens}^{SRF} > 2 S_{Sens}^{Pixel},$$

clearly stating that the image of a physical object can be perfectly reconstructed by a digital sensor, if the size of the image's SRF on the sensor is at least twice the size of the sensor's pixels.

³⁰A signal is said to be (*absolutely*) *bandlimited*, if there exist two finite frequencies f_{\min} and f_{\max} , such that the signal has no spectral components outside the interval $[f_{\min}, f_{\max}]$ (Couch II, 2013). The term *highest frequency content* of a signal refers to the maximum frequency f_{\max} , and is usually denoted by B .

Note that, as Gonzalez and Woods, 2008 emphasize, perfect recovery is ensured only if the sampling rate *exceeds* the Nyquist frequency. On a similar note, a more conservative criterion suggests that since the sampling theorem was originally formulated for 1D signals, it does not sufficiently accommodate the needs of signals in 2D. According to this criterion (see e.g. Andor Oxford Instruments, 2017), a faithful representation of objects can only be achieved if the size of the SRF on the sensor is *at least three times* the size of the sensor's pixels, i.e.

$$S_{Sens}^{SRF} \geq 3 S_{Sens}^{Pixel}. \quad (4.25)$$

Equations 4.19 – 4.23 and 4.25 form the basis for the subsequent analysis.

For a given working distance W and thickness factor ℓ_{TH} , the size S_{Print}^{NLC} of the printed NLCode should not be so small that the SRF is not adequately resolved. The minimum print size $S_{Print,min}^{NLC}$ is found from Eqs. 4.19, 4.20, 4.23 and 4.25, by taking S_{Sens}^{SRF} equal to $3 S_{Sens}^{Pixel}$, i.e. adopting the conservative version of the Nyquist criterion, and making the appropriate substitutions and rearrangements of terms:

$$S_{Print,min}^{NLC} = \frac{3 S_{Sens}^{Pixel}}{f \cdot \ell_{TH}} \cdot W \leq S_{Print}^{NLC}. \quad (4.26)$$

In order to prevent erroneous clippings of the NLCode's image, S_{Print}^{NLC} should also not be too large for the given working distance; for the purpose of this calculation, W is assumed *fixed* and less flexible than the feature's size. Using Eqs. 4.20 and 4.21, the NLCode's maximum print size $S_{Print,max}^{NLC}$ is found to be

$$S_{Print,max}^{NLC} = \frac{\min [N_{Pixel}^H, N_{Pixel}^V] \cdot S_{Sens}^{Pixel}}{f} \cdot W \geq S_{Print}^{NLC}. \quad (4.27)$$

Equations 4.26 and 4.27 can be rearranged in order to provide the maximum W_{max} and minimum W_{min} working distances for a given print size S_{Print}^{NLC} and ℓ_{TH} , that is,

$$W_{max} = \frac{f \cdot \ell_{TH}}{3 S_{Sens}^{Pixel}} \cdot S_{Print}^{NLC} \geq W \quad (4.28)$$

and

$$W_{min} = \frac{f}{\min [N_{Pixel}^H, N_{Pixel}^V] \cdot S_{Sens}^{Pixel}} \cdot S_{Print}^{NLC} \leq W, \quad (4.29)$$

respectively. The physical meanings of W_{max} and W_{min} are analogous to those of their size counterparts. Specifically, W_{max} is the maximum working distance

from an NLCode of print size S_{Print}^{NLC} that will allow the SRF to be adequately defined on the captured image, and W_{min} the minimum distance that will prevent the cropping of the same NLCode's image.

W_{max} and W_{min} are subject to two conditions, each of which provides more insight into the parameters involved in the specification of the NLCode's OW.

1. The first condition simply states the obvious – but not trivial – fact that the minimum working distance must be less than or equal to the maximum working distance, i.e.

$$W_{min} \leq W_{max} .$$

Making the appropriate substitutions to the left and right hand side of the above inequality defines an *absolute lower threshold* $\ell_{TH,max}^{Abs}$ for the thickness factor as

$$\ell_{TH,min}^{Abs} = \frac{3}{\min [N_{Pixel}^H, N_{Pixel}^V]} \leq \ell_{TH} . \quad (4.30)$$

If the denominator in the equation above is set to the approximate value of 3000 pixels (see [Table 4.1](#)), then the lower threshold of ℓ_{TH} is estimated to be

$$\ell_{TH,min}^{Abs} \simeq 10^{-3} . \quad (4.31)$$

Note that the term *absolute* is generally used to indicate that the threshold in question – minimum, maximum – does not depend on one or both of the two main variables S_{Print}^{NLC} and W , nor does it depend on the parameter ℓ_{TH} – all subject to the threshold defined and the relevant context.

2. The second condition has already been discussed. It relates to the second issue raised when W becomes too small; the camera is unable to properly focus on the NLCode. Since according to [Eq. 4.22](#) the working distance must always be greater than or equal to the minimum focusing distance W_{min}^{Abs} , so does W_{min} given by [Eq. 4.29](#). The inequality

$$W_{min} \geq W_{min}^{Abs} , \quad (4.32)$$

defines the *absolute minimum print size* $S_{Print,min}^{NLC,Abs}$ of the NLCode

$$S_{Print,min}^{NLC,Abs} = \frac{3 S_{Sens}^{Pixel}}{f \cdot \ell_{TH,min}^{Abs}} \cdot W_{min}^{Abs} \leq S_{Print}^{NLC} \quad (4.33)$$

which, for the known values of all the parameters involved is found to be

$$S_{Print,min}^{NLC,Abs} \simeq 4.44 \text{ cm} . \quad (4.34)$$

Like the limiting values of the working distances, $S_{Print,min}^{NLC}$ and $S_{Print,max}^{NLC}$, that is, the minimum and maximum print sizes of the NLCode respectively, are subject to analogous conditions:

1. The fact that the minimum print size must not be larger than the maximum print size of the NLCode, i.e.

$$S_{Print,min}^{NLC} \leq S_{Print,max}^{NLC} ,$$

simply reproduces Eq. 4.30 concerning the absolute minimum value of ℓ_{TH} .

2. Before subjecting the working distances to their respective conditions, there was no absolute minimum value for the print size of the NLCode. Equation 4.33 changed that, so now $S_{Print,min}^{NLC}$ must also be

$$S_{Print,min}^{NLC} \geq S_{Print,min}^{NLC,Abs} . \quad (4.35)$$

This condition sets an upper limit to the thickness factor ℓ_{TH} , for a given working distance W , that is,

$$\ell_{TH,max} = \frac{\ell_{TH,min}^{Abs}}{W_{min}^{Abs}} \cdot W \geq \ell_{TH} . \quad (4.36)$$

This limit simply – although not straightforwardly – enforces the “hard” limits $S_{Print,min}^{NLC,Abs}$ and W_{min}^{Abs} set on the print size of the NLCode and the distance between the camera and the feature respectively. The main observation is that if, for any given W , ℓ_{TH} is allowed to become greater than $\ell_{TH,max}$, then Eq. 4.35 might be violated, since the increased ℓ_{TH} will decrease $S_{Print,min}^{NLC}$ (Eq. 4.26), without affecting $S_{Print,min}^{NLC,Abs}$ (Eq. 4.33). Ignoring the existence of $S_{Print,min}^{NLC,Abs}$, one might create an NLCode of size less than that. In such an event, even if one uses the raised value of ℓ_{TH} , and therefore preserve a value of L_{Print}^{TH} that meets the condition set by Eq. 4.23, i.e. ensure that the lines of the pattern are adequately defined, that smaller size of the NLCode will cause a decrease in W_{min} – possibly making it less than W_{min}^{Abs} , thus creating the potential to face a focusing problem. Moreover, if one creates a small NLCode but using a low ℓ_{TH} , L_{Print}^{TH} might become less than S_{Print}^{SRF} , adding low definition to an already poor setup. In other words, this last condition set on ℓ_{TH} ties up the one loose end that could jeopardise the consistency of this analysis.

Determining an absolute upper limit for ℓ_{TH} is not as straightforward. The requirement that $L_{Print}^{TH} \geq S_{Print}^{SRF}$ (Eq. 4.23) has already led to the definition of $S_{Print,min}^{NLC}$ (Eq. 4.26), and it would be trivial – to say the least – and of no practical use to suggest that $S_{Print,min}^{NLC} \geq S_{Print}^{SRF}$. The upper threshold for ℓ_{TH} can only be determined iteratively, i.e. through trial and error, by the NLCode Generator which tests different trajectories against criteria based on spatial considerations. The thickness factor ℓ_{TH} was introduced into the NLCode scheme exactly in order to render the Generator independent of the intended print size S_{Print}^{NLC} of the feature. Therefore, regardless of the intended print size of the NLCode, by increasing ℓ_{TH} from some small value, the Generator will start failing an increasing number of trajectories, and eventually reach a point – for some critical value of ℓ_{TH} – where it can no longer find any trajectories that meet the criteria. The upper threshold for the thickness factor should be set lower than that critical value. Note that since this process is independent of the size of the NLCode, and therefore also independent of any intended working distance, this upper threshold is an absolute one, and for this reason is denoted by $\ell_{TH,max}^{Abs}$. A search based on the description just given suggested that

$$\ell_{TH,max}^{Abs} \simeq 10^{-2} \quad (4.37)$$

is a good estimate for this upper threshold.

For a given camera, i.e. for known, fixed values of f , S_{Sens}^{Pixel} , N_{Pixel}^H , and N_{Pixel}^V , as well as some preset value for ℓ_{TH} , Eq. 4.26, 4.27, and Eq. 4.29, 4.28 define a *linear dependence* of

- The minimum and maximum print size of the NLCode, on the working distance and
- The maximum and minimum working distance, on the print size of the NLCode respectively.

Note that (a) while the thickness factor enters the definitions of $S_{Print,min}^{NLC}$ and W_{max} as a parameter, $S_{Print,max}^{NLC}$ and W_{min} do not depend on ℓ_{TH} , and (b) the slope of $S_{Print,min}^{NLC}$ as a function of W is *inversely proportional* to ℓ_{TH} , and the slope of W_{max} as a function of S_{Print}^{NLC} is *proportional* to ℓ_{TH} .

Taking the values of the parameters entering Eqs. 4.26 and 4.27 from Table 4.1, the quantities $S_{Print,min}^{NLC,Abs}$, $S_{Print,min}^{NLC}(\ell_{TH}; W)$, and $S_{Print,max}^{NLC}(W)$ are plotted in logarithmic scales – such that the slopes of the straight lines plotted are indicated by the ordinates of the respective lines – and shown in Fig. 4.14. This figure

describes a situation in which, given a specific working distance W , one inquires as to the minimum and maximum allowed print sizes of the NLCode, as well as the maximum thickness factor $\ell_{TH,max}$ (Eq. 4.36).

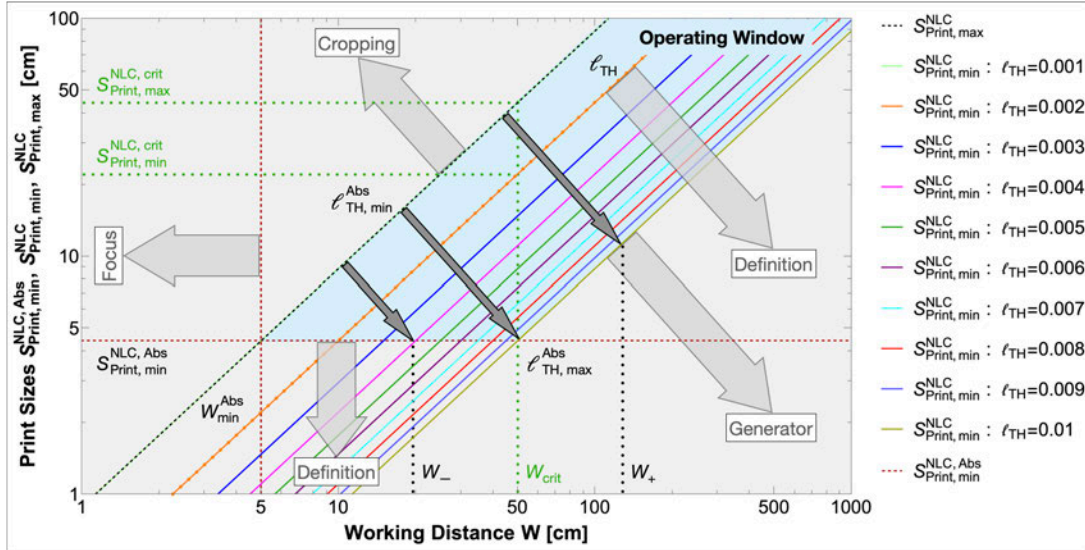


Figure 4.14: NLCode's Operating Window: Print size Vs Working distance: Absolute minimum $S_{Print,min}^{NLC,Abs}$, minimum $S_{Print,min}^{NLC}(\ell_{TH}; W)$ and maximum $S_{Print,max}^{NLC}(W)$ allowed print sizes of the NLCode, the second parametrised by the thickness factor ℓ_{TH} – as indicated in the legend to the right, and the latter two as functions of the working distance W . The light blue shaded area marks the NLCode's OW in terms of print size and working distance. The absolute upper threshold $\ell_{TH,max}^{Abs}$ imposed on ℓ_{TH} by the NLCode Generator creates a critical working distance W_{crit} (Eq. 4.38) which sets the upper limit of the thickness factor to $\ell_{TH,max}(W_-)$ (Eq. 4.36) for $W = W_- \leq W_{crit}$, and to a fixed $\ell_{TH,max}^{Abs}$ (Eq. 4.37) when $W = W_+ > W_{crit}$. The dark grey arrows show the range of allowed values of the thickness factor, in each case. The minimum print size of the NLCode is given by Eq. 4.26 in all cases, simply becoming $S_{Print,min}^{NLC,Abs}$ (Eq. 4.33) when $W < W_{crit}$. The grey areas surrounding the OW represent the forbidden ranges of the parameters involved. Each of those areas is marked by a light grey arrow whose beginning indicates the limiting value of the relative parameter, and its end leads to a descriptor of the ultimate cause of the limitation. For example, the descriptor Definition refers to the print size becoming too small for the SRF to be properly defined (Eq. 4.23). Given a fixed value of ℓ_{TH} within the feature's OW – in the example shown in orange, $\ell_{TH} = 0.002$ – and an intended working distance, say $W = W_{crit}$, the minimum and the maximum allowed print sizes of the NLCode are found from Eqs. 4.26 and 4.27 respectively.

According to the preceding analysis, Eqs. 4.26, 4.27 and 4.36 should straightforwardly provide the answer to the above inquiry. However, the absolute upper threshold $\ell_{TH,max}^{Abs}$ (Eq. 4.37) is not a result of the analysis, but rather an additional restriction imposed by the NLCODE Generator. In Fig. 4.14, this restriction is shown in the lines representing $S_{Print,min}^{NLC}(\ell_{TH}; W)$ for various values of ℓ_{TH} . If the thickness factor was allowed to increase beyond $\ell_{TH,max}^{Abs}$, those lines would extend to lower ordinates – remember the inverse proportionality between $S_{Print,min}^{NLC}(\ell_{TH}; W)$'s slope and ℓ_{TH} – without limitation. Because of $\ell_{TH,max}^{Abs}$, however, this is not the case. The effect of this “externally” imposed threshold is that, through the intersection of the lines representing $S_{Print,min}^{NLC,Abs}$ and $S_{Print,min}^{NLC}(\ell_{TH,max}^{Abs}; W)$, it creates a *critical value* W_{crit} of the working distance, given by

$$W_{crit} = \frac{\ell_{TH,max}^{Abs}}{\ell_{TH,min}^{Abs}} \cdot W_{min}^{Abs}. \quad (4.38)$$

For $W = W_- \leq W_{crit}$, the maximum thickness factor is $\ell_{TH,max}(W_-)$, given by Eq. 4.36, and the print size's lower threshold is $S_{Print,min}^{NLC,Abs}$, given by Eq. 4.33 (also by Eq. 4.26 for the appropriate parameter values). When $W = W_+ > W_{crit}$, the maximum thickness factor is fixed to $\ell_{TH,max}^{Abs}$, i.e. it stops following Eq. 4.36, and the print size's lower threshold is given by Eq. 4.26.

Figure 4.15 describes the opposite situation, where given a specific print size S_{Print}^{NLC} , one inquires as to the minimum and maximum allowed working distances of the NLCODE, as well as the maximum thickness factor $\ell_{TH,max}$ (Eq. 4.36). This case is much simpler than the previous one, since the absolute threshold $\ell_{TH,max}^{Abs}$ imposed by the Generator does not create any critical values of the print size – analogous to W_{crit} of the case earlier examined. The reason for this is that this time, the thickness factor ℓ_{TH} parametrises the maximum working distance $W_{max}(\ell_{TH}; S_{Print}^{NLC})$ (Eq. 4.28), which is not linked to any “hard” limits of the parameters involved – as was the case with $S_{Print,min}^{NLC}(\ell_{TH}; W)$. Provided the desired print size remains above the absolute minimum threshold $S_{Print,min}^{NLC,Abs}$ given by Eq. 4.33, the thickness factor is free to take any value within its absolute range given by Eqs. 4.31 and 4.37, and the range of allowed working distances will be defined by Eqs. 4.28 and 4.29, after the appropriate parameter assignments.

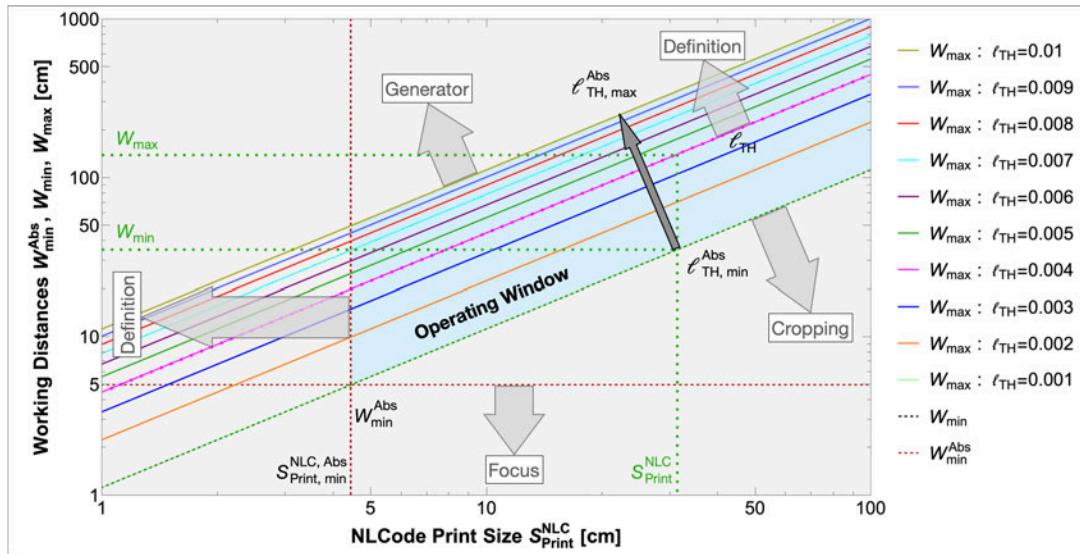


Figure 4.15: NLCODE's Operating Window: Working distance Vs Print size: Absolute minimum W_{min}^{Abs} , minimum $W_{min}(S_{Print}^{NLC})$ and maximum $W_{max}(\ell_{TH}; S_{Print}^{NLC})$ allowed working distances, the latter parametrised by the thickness factor ℓ_{TH} – as indicated in the legend to the right, and the latter two as functions of the print size S_{Print}^{NLC} of the NLCODE. The light blue shaded area marks the NLCODE's OW in terms of working distance and print size. Given a fixed value of ℓ_{TH} within the feature's OW – in the example shown in magenta, $\ell_{TH} = 0.004$ – and an intended print size S_{Print}^{NLC} , the minimum and the maximum allowed working distances are found from Eqs. 4.28 and 4.29 respectively. The thickness factor in this case is free to take any value within its absolute range given by Eqs. 4.31 and 4.37. This figure is colour-coded and marked the same as Fig. 4.14.

Part II

Results & Discussion

It is however certain, that no estimate is more in danger of erroneous calculations than those by which a man computes the force of his own genius.

SAMUEL JOHNSON,
Rambler, No. 154 (Murphy, 1840)

5

The NLCode Reader via a Demo App

5.1 NLCode Capture: Target Area



Figure 5.1: Demo App - Camera View with Target Area On: The first view of the Demo App shows the current scene from the camera's viewpoint. The Target Area Button (top-left) is a toggle switch which controls the UIE called target area. The target area is a semi-transparent view with a transparent ring at its centre, whose purpose as a UIE is to help the user take a "good" picture of the NLCode, i.e. a picture that can be used by the processes of the Demo App following capture. The Torch Button (top-centre) is another toggle switch which turns the torch of the iPhone on and off. The torch can, in some cases, improve the lighting conditions and therefore the feature's performance. The user also has the option to scan a picture of the NLCode previously taken, by tapping the Photo Library Button (top-right). The Capture Button (bottom-centre) has the obvious main use of the view, which is to take a picture of the NLCode aimed at. Lastly, a Focus Indicator (not shown), is an animated "blinking" circle which appears when the user taps in the area they wish the camera to focus on.

Figure 5.1 shows the first view of the Demo App. This view is called *camera view* because it displays the current scene from the camera's viewpoint. In the instance shown, the iPhone's primary (rear) camera (Table 4.1), hereon simply referred to as *camera*, is pointed at an NLCode, which was printed on regular matte paper using a consumer inject printer. The *User Interface (UI)* on the camera view presents the user with the options available to them. Each option corresponds to one of the *User Interface Elements (UIEs)* described below.

■ **Top View (left to right):**

- **Target Area Button** – All *User Interface Buttons (UIBs)* are *control elements*, and a large majority of them are called *toggle switches* because they operate *two states* of the function they perform, or the UIE they control. The Target Area Button is a toggle switch, and its function is to insert/remove another view into/from the camera view. This overlay view contains a very important element of the Demo App called *target area*, which serves as more than a UIE, and whose purpose will be discussed in this section, as well as in Subsecs. 5.2.3 and 5.2.4. The Target Area Button takes its name from that element, and its icon is a *target* or, more appropriately, a *reticle*³¹.
- **Torch Button** – The Torch Button is also a toggle switch, which turns the torch of the iPhone on and off. The use of colours in the NLCode scheme makes it very sensitive to lighting conditions and an illuminated scene can counteract both low light conditions, and interference from ambient lighting or other light sources (see Subsec. 5.2.2). The icon of the Torch Button is a flashlight which is crossed out when the torch is off.
- **Photo Library Button** – The Photo Library Button gives the user the option to use a picture of the NLCode previously taken. Tapping this button opens the *Photos App* on the user's device and provides access to their Photo Library. The Photo Library Button is mostly a developer's tool, which is necessary in order to reproduce bugs and apply fixes during development. The icon of the Photo Library Button is a standard system symbol of a photo stack.

³¹ A *reticle*, or *reticule* is "A grid or other pattern of fine threads, wires, lines, etc., in the focal plane or eyepiece of a telescope or other optical instrument in order to facilitate positioning, aiming, and measurement. Also called *reticule*. Cf. *graticule* n. 2." (reticle - OED, 2023).

- **Bottom View (centre):**
 - **Capture Button** – The main control UIE of the camera view is the Capture Button – technically a *gesture recogniser* – which enables the user to take a picture of an NLCode, if they choose to. The Capture Button’s icon is a camera aperture.
- **Anywhere in the Camera View:**
 - **Focus Indicator (not shown)** – The Focus Indicator is another gesture recogniser, which enables the user to indicate the area of the scene they wish the camera to focus on. When the user taps anywhere in the Camera View, the Focus Indicator appears as a “*blinking*” animated circle around the point tapped.

The view controlled by the Target Area Button is a semi-transparent white layer with a *transparent ring* in its center (Fig. 5.1). This ring will mostly be referred to as *target area*, as earlier mentioned, but it can also be called *scanner reticle*, or *scanner cutout*. The general purpose of the target area is to define the area in which the arced circular frame is expected to be in the captured image of an NLCode. As a UIE, the scanner reticle helps the user position the camera at an appropriate distance from the feature, *and* place the NLCode at the center of the view, before taking a picture; it therefore fits the “*positioning-aiming-measurement*” part of a reticle’s description (footnote 31). From a *back-end’s* perspective, the target area forms the basis a series of three consecutive processes are build upon. This function of the target area is discussed in detail in Subsecs. 5.2.1, 5.2.3 and 5.2.4.

The size of the ring is defined by its *outer* and *inner radius* R_{outer}^{TA} and R_{inner}^{TA} respectively, where the superscript *TA* stands for “Target Area”. In order to find suitable values for the two radii, one must first take into account the different *coordinate spaces* utilised by Apple’s *native drawing technologies* and *system frameworks*. According to the company’s *Documentation Archive* for iOS developers (Apple Inc., 2012), views, and therefore any native technology used for drawing on them, make use of a *logical coordinate space*, in which lengths are measured in *Logical Points (LPs)*. The target area is a graphics object drawn on a view of the Demo App, which means that R_{outer}^{TA} and R_{inner}^{TA} must be specified in LPs. The mapping of logical points to pixels is done by system frameworks, which use the device’s *pixel coordinate space*. Since each device has different screen resolution, system frameworks are tasked with calculating a property of graphics objects

called *scale factor*, which the drawing technologies require as input in order to create those objects. The scale factor, denoted here by $f_{Pixels/LP}$ for clarity, is the number of pixels that correspond to one logical point.

Formally, the scale factor is calculated based on the device's *screen resolution* – not its *camera resolution*, which means that in the above definition of $f_{Pixels/LP}$, the term *pixels* refers to *screen pixels*. This is because most applications are interested in rendering objects on the device's screen – not drawing them on the camera scene with the intention to later apply them on an image of the scene obtained through capture. The latter is, however, exactly what the Demo App needs to do. For this reason, the scale factor is defined here as the the number of *camera* or *sensor pixels* that correspond to one LP.

Apart from the camera resolution, which is given in [Table 4.1](#), in order to calculate $f_{Pixels/LP}$ one also needs to know the *logical resolution* ($N_{LP}^H \times N_{LP}^V$), i.e. the number of logical points along the horizontal and vertical dimensions of the views implemented in the Demo App, denoted by N_{LP}^H , and N_{LP}^V respectively. The logical resolution of the Demo App is

$$(N_{LP}^H \times N_{LP}^V) = (375 \times 667) \text{ LPs.} \quad (5.1)$$

Having different resolutions along the horizontal and vertical dimensions means that the scale factor can be calculated based on either one of them – as long as both N_{Pixel} and N_{LP} used in the calculations refer either to the horizontal, or the vertical dimension. Since the Demo App only operates in *portrait mode*, the smallest – camera *and* logical – resolution is along the horizontal dimension. Therefore, the scale factor is defined with respect to the horizontal dimension, i.e.

$$f_{Pixels/LP} = \frac{N_{Pixel}^H}{N_{LP}^H} = \frac{3024}{375}. \quad (5.2)$$

The target area is defined as a ring, instead of simply a disk within which the NLCode is to be placed during capture, for reasons best explained by considering the specific purpose served by the ring's outer and inner circular edges. The target area silently – yet intuitively – prompts the user to move the camera closer, or further away from the NLCode, in order to make sure the arced circular frame depicted on the scene is *smaller* than the outer edge, and *larger* than the inner edge of the ring. On the one hand, moving the camera closer to the NLCode increases the feature's size on the scene, and risks having the arced circular frame fall outside the outer edge of the ring. On the other hand, increasing the distance

between the camera and the NLCode makes the feature appear smaller, and risks having the arced circular frame become smaller than the inner edge of the ring. A suitably chosen outer radius R_{outer}^{TA} can protect against *cropping* and/or *focusing* problems, by preventing the user from moving the camera too close to the feature. Correspondingly, a suitably chosen inner radius R_{inner}^{TA} stops the user from moving the camera too far from the feature, which can cause issues with definition.

The smallest NLCode print size plays a crucial role in both situations, since it is a small feature that is associated with the user's urge to move the camera closer to it, and at the same time, it is the small feature that has a higher risk of facing definition issues when captured from large distances. Figure 4.15 of Sec. 4.4 shows that for an NLCode of the absolute smallest print size, i.e. $S_{Print,min}^{NLC,Abs}$ (Eq. 4.33), the camera should not be at a distance less than W_{min}^{Abs} to prevent cropping and focusing issues, and at a distance greater than $W_{max}(\ell_{TH,max}^{Abs}; S_{Print,min}^{NLC,Abs}) = W_{crit}$ (Eq. 4.38) to prevent definition issues. Using the pinhole camera model (Eq. 4.20), the two (*print size, working distance*) pairs $(S_{Print,min}^{NLC,Abs}, W_{min}^{Abs})$ and $(S_{Print,min}^{NLC,Abs}, W_{crit})$, admit two different sizes of the same NLCode *on the sensor*, since

$$S_{Sens}^{NLC} = f \cdot \frac{S_{Print}^{NLC}}{W}. \quad (5.3)$$

This in turn leads to two different NLCode pixel sizes *on the captured image*, since

$$S_{Pixel}^{NLC} = \frac{S_{Sens}^{NLC}}{S_{Sens}^{Pixel}}, \quad (5.4)$$

where S_{Sens}^{Pixel} is the physical size of a pixel on the camera sensor (Table 4.1). Finally, the two different pixel sizes of the NLCode correspond to two different sizes of the feature *on the views* implemented by the Demo App, which are measured in LPs and calculated using the scale factor $f_{Pixels|LP}$ given by Eq. 5.2, that is,

$$S_{LP}^{NLC} = \frac{S_{Pixel}^{NLC}}{f_{Pixels|LP}}. \quad (5.5)$$

By making a few obvious substitutions, the three above Eqs. 5.3 – 5.5 can be combined to a single equation, that is,

$$S_{LP}^{NLC} = \frac{f}{S_{Sens}^{Pixel} \cdot f_{Pixels|LP}} \cdot \frac{S_{Print}^{NLC}}{W}, \quad (5.6)$$

which directly links pairs of print size and working distance to NLCode sizes in LPs, using a few known camera specifications.

The first pair, i.e. $(S_{Print,min}^{NLC,Abs}, W_{min}^{Abs})$, defines through Eq. 5.6 the *diameter* $2R_{outer}^{TA}$ of the target area's outer edge; it is the maximum size the NLCODE is allowed to have on the view. Not surprisingly, $2R_{outer}^{TA} = 375 LPs$, i.e. it is equal to the horizontal logical resolution N_{LP}^H of the Demo App views (Eq. 5.1) – after that, cropping and focusing issues ensue. Correspondingly, the second pair, that is, $(S_{Print,min}^{NLC,Abs}, W_{crit})$, defines the *diameter* $2R_{inner}^{TA}$ of the target area's inner edge; it is the minimum size the NLCODE is allowed to have on the view. $2R_{inner}^{TA} \simeq 37.5 LPs$, i.e. approximately one order of magnitude smaller than $2R_{outer}^{TA}$, which accounts for the range of ℓ_{TH} 's values – after that, definition issues ensue.

Letting $R_{outer}^{TA} = 375/2 LPs$ and $R_{inner}^{TA} \simeq 37.5/2 LPs$ may be consistent with the OW of the NLCODE, and will certainly provide proper user guidance. However, the target area serves purposes not yet discussed, which require a better definition of the arced circular frame's placement on the Demo App views and ultimately, on the captured image. For this reason, both R_{outer}^{TA} and R_{inner}^{TA} are used in conjunction with the arced circular frame specifications (see Sec. 3.1), to each provide its own estimation for R_{inner}^{TA} and R_{outer}^{TA} respectively. This leads to two pairs of outer and inner radii for the target area, i.e.

$$R_{outer}^{TA} = 375/2 = 187.5 LPs \xrightarrow{\text{Frame's Specs}} R_{inner}^{TA} = 125 LPs \quad (5.7a)$$

$$R_{inner}^{TA} \simeq 37.5/2 = 18.75 LPs \xrightarrow{\text{Frame's Specs}} R_{outer}^{TA} \simeq 28 LPs. \quad (5.7b)$$

The final size of the target area is found by taking the *mean* of each pair of corresponding values and rounding to a suitable multiple of ten, leading to

$$R_{outer}^{TA} = 100 LPs \quad (5.8a)$$

$$R_{inner}^{TA} = 60 LPs. \quad (5.8b)$$

Note that the above values for the outer and inner radii of the target area are also an intuitive choice based on the developer's experience with the Demo App, that has been seen to allow the successful capture and processing of NLCODEs under various settings permissible by the feature's OW. The above analysis substantiates that choice, and provides a roadmap of the reasoning behind it. The target area shown in Fig. 5.1 was created according to the specifications given in Eqs. 5.8.

5.2 Image Processing

As soon as the NLCode aimed at is captured, the Demo App transitions to a second view called *initial photo view* (Fig. 5.2), because it displays the photograph of the feature just taken. This view also presents the user with a few options (see Subsec. 5.2.1), one of which is to tap onto a newly presented UIB in order to initiate the successive application of five image processing techniques. These methods are the subject of the present section, and include *Masking*, *Colour Detection*, *Corner Detection & Identification*, *Rotation*, and *Perspective Correction*.

5.2.1 Masking

Figure 5.2 shows the second view of the Demo App. This view is called *initial photo view* because it displays the photograph of the NLCode previously captured (or taken from the Photo Library). The UI on the initial photo view presents the user with a few options, each corresponding to a UIE displayed on the view.

■ Top View:

- **Target Area Button (left)** – The Target Area Button (Sec. 5.1) remains available to the user after capture, offering them a chance to switch the target area on and inspect the positioning of the feature relative to the scanner reticle.
- **Save Button (right)** – The Save Button gives the user the option to save the photo of the NLCode just taken, to their Photo Library. The icon of the Save Button is a standard system symbol of a downward arrow entering a square.

■ Bottom View:

- **Back Button (left)** – If, for whatever reason, the user wishes to retake a picture of the NLCode, they can do so by tapping the Back Button. This UIE instructs the Demo App to display the camera view and initiate a new camera session. The Back Button's icon is a standard system symbol of a backward arrow.
- **Apply Masks Button (right)** – The main control UIE of the initial photo view is the Apply Masks Button, which enables the user to initiate the processing of the NLCode. The first image processing technique applied on the captured image of the feature is *masking*, which is the main topic of the present subsection. The Apply Masks Button's icon simply displays the text *Apply Masks*.



Figure 5.2: Demo App - Initial Photo View & Masking prompt: The second view of the Demo App shows the captured photograph of the NLCode. The target Area Button is still displayed on the top-left of the view, giving the user a chance to switch the target area on, in order to inspect the captured image in terms of the feature placement in it. The Save Button at the top-right of the view offers the user the option to save the image to their Photo Library. If the user is not pleased with the result, they can move back to the camera view in order to retake the picture, by tapping the Back Button (bottom-left). The Apply Masks Button displayed at the bottom-right of the view, prompts the user to proceed with the processing of the NLCode image, when they are ready to do so. The masks applied on the captured image when the Apply Masks Button is tapped, separate the arced circular frame from the nonlinear pattern by creating two new images, each containing only one of the two elements of the feature.

Masking is an image processing technique used to isolate specific areas on an image, and/or separate them from other areas. A most common example of image masking is background removal. Using a variety of selection criteria, oftentimes based chromatic and/or spatial characteristics of an image, background removal techniques identify and separate the subject from its background. This is usually done in a non-destructive way, using mask layers in order to keep the original image intact, but masks can also be applied in a destructive way, by permanently altering the pixels of the original image. The masks discussed in this subsection

are applied on the original image of an NLCode, and their purpose is to separate the arced circular frame from the nonlinear pattern of the feature, based on spatial criteria that make use of the target area.

The target area acts as a guide for placing the arced circular frame inside its transparent ring, and in effect, causes the spatial separation between the frame and the nonlinear pattern. However, as can be seen in Fig. 5.1, due to the inevitable perspective projection applied on the image, the two elements of the feature are not completely separated – a small part of the pattern lies inside the ring. Provided the inner quiet area of the NLCode is sufficiently wide (Table 3.1), this issue can be remedied by increasing the inner radius R_{inner}^{TA} . The method used for the calculation of the *maximum inner radius of the target area*, that makes sure no parts of the pattern are inside the ring without cutting out any parts of the frame, is described below.

A circle drawn on a digital image consists of a list of pixels which, given the centre and the radius of the circle (in pixels), can be calculated using a computer graphics technique known as *midpoint circle algorithm* (Hearn et al., 2010).

The ring of the target area is located around the centre of the Demo App's views, which corresponds to the centre C_{img} of the processed images of the NLCode. The radii of the target area are known in LPs (Eqs. 5.8), but can be converted to pixels using equation

$$R_{inner/outer}^{TA, Pixel} = f_{Pixels/LP} \cdot R_{inner/outer}^{TA, LP}, \quad (5.9)$$

where $f_{Pixels/LP}$ is the scale factor introduced in Sec. 5.1 (Eq. 5.2), and the superscripts *Pixel* and *LP* on the target area radii introduce an obvious distinction. Based on the above remarks, the centre and the radii of the target area – the latter rounded down to the nearest multiple of ten, are, in pixels,

$$C_{img} = (1512, 2016) \text{ Pixels} \quad (5.10a)$$

$$R_{outer}^{TA, Pixel} = 800 \text{ Pixels} \quad (5.10b)$$

$$R_{inner}^{TA, Pixel} = 480 \text{ Pixels}. \quad (5.10c)$$

The maximum inner radius $R_{inner, max}^{TA, Pixels}$ of the target area that does not allow the ring's inner edge to touch the arced circular frame, is found via an iterative process. Using the midpoint circle algorithm to compute the pixels forming a circle centred at C_{img} , with a radius that starts from $R_{inner}^{TA, Pixel}$ and increases by a certain

number of pixels ΔR , say 10, in every iterative step, this process accesses the pixels of the NLCode's image, looking for pixels that belong to the arced circular frame. The maximum inner radius $R_{inner, max}^{TA, Pixels}$ is defined as the radius of the largest circle that does not contain any frame pixels.

Frame pixels are identified based on their brightness value b , with respect to the *HSB colour space*. When the brightness of a pixel is below a certain (low) brightness threshold b_{th} , the pixel is considered a frame pixel. The process identifying the appropriate threshold b_{th} is part of the analysis relating to colour detection, which is presented in [Par. 5.2.2.2 \(Table 5.1\)](#).

The iterative process approaching the arced circular frame from the inner edge of the target area's ring, is also used in order to approach the frame from the outside, i.e. using $R_{outer}^{TA, Pixel}$ as the starting radius, and decreasing it by ΔR in every iterative step. This results to the *minimum outer radius of the target area*, $R_{outer, max}^{TA, Pixels}$, that does not allow the ring's outer edge to touch the arced circular frame, and ensures that any objects surrounding the feature will not interfere with the reading process.

[Figure 5.3](#) shows the original image of the NLCode, initially masked by an image with green background and a transparent ring at its centre, with inner and outer radii equal to those of the target area displayed on the camera view ([Fig. 5.1](#)). The raster circles with the maximum inner and minimum outer radii found by the iterative process described above, were calculated using the midpoint circle algorithm, and drawn on the first masked image (left of [Fig. 5.3](#)), in blue and red pixels respectively. The values of the radii are given in the figure's main caption. Note that the two circles closing in on the frame will be different for every captured image, of even the same NLCode. The second and third images show the same image of the NLCode, but this time masked by two different images; one with a transparent *ring* using the calculated radii, that only exposes the arced circular frame (centre of [Fig. 5.3](#)), and one with a transparent *disk* of radius $R_{inner, max}^{TA, Pixels}$, that only exposes the nonlinear pattern of the feature (right of [Fig. 5.3](#)).

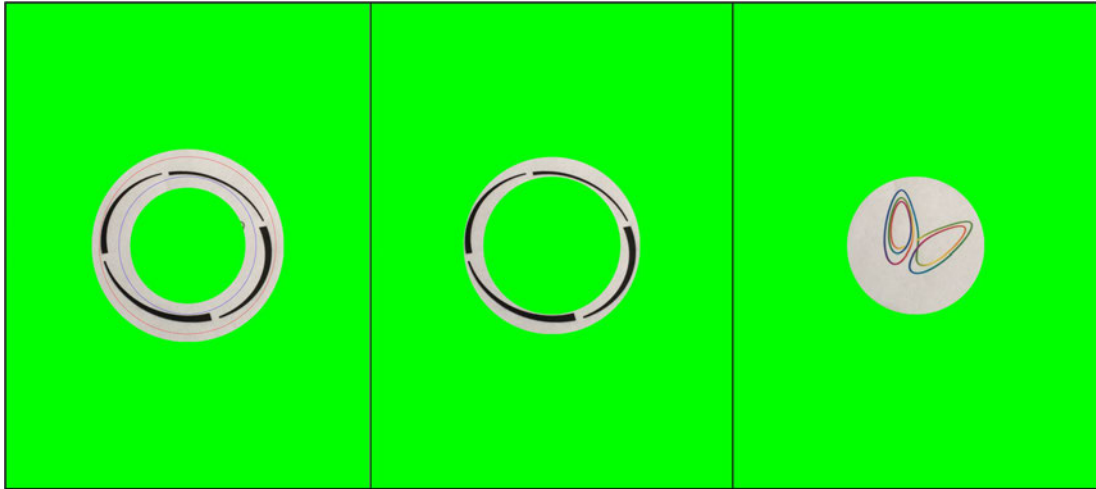


Figure 5.3: Masking of the NLCode's captured image: The image on the left is the photograph of the NLCode captured, masked by an image of green background, with a transparent ring at its centre, which has the size of the target area displayed on the camera view (Fig. 5.1, Eqs. 5.10). The radii of the blue and red raster circles drawn on that image are the maximum inner and minimum outer radii calculated using the iterative process described in this subsection, and their values are $R_{inner,max}^{TA, Pixels} = 571$ Pixels and $R_{outer,max}^{TA, Pixels} = 735$ Pixels respectively. These radii were computed in order to define the circular elements of the two masks created for the separation of the frame from the nonlinear pattern of the NLCode. The image in the center of the figure shows the same image of the feature, but this time masked by an image with a transparent ring at its centre, whose radii are those calculated iteratively. This mask exposes the frame of the NLCode, and blocks all other elements behind its green background. The image on the right was masked by an image with a transparent ring at its centre, and radius $R_{inner,max}^{TA, Pixels}$. This image isolates the nonlinear pattern and blocks every other elements of the feature.

5.2.2 Colour Detection & Filtering

This subsection presents the development process of the *output palette* of the NLCode's colour profile, starting with the *test photographic sample* created for that purpose. Like the input palette presented in section Sec. 4.3, the output palette is a sequence of distinct colours. Unlike the input palette, however, the output palette is not a feature of the NLCode that can be predefined. It consists of a certain number of colours a camera *detects* on the line representing the nonlinear pattern of the feature, and for this reason, it needs to be developed through the processing of a sample of the NLCode's photographs taken under

different conditions. Moreover, if the output palette developed is to assist the tracing process implemented by the NLCode Reader (Sec. 5.3), thus fulfilling the colour profile's secondary purpose mentioned in Sec. 3.4, it must have certain properties. These properties are discussed in detail in Par. 5.2.2.3.

5.2.2.1 Test Photographic Sample

The study performed on the test sample presented in this paragraph, aims at determining whether the intended use of colours in the NLCode scheme is a viable option or not. For this reason, rather than placing emphasis on the exact specifications of the NLCodes photographed, the precise measurement of the working distance and lighting conditions, and the size of the sample, the test photographic sample presented here, focuses on exploring a minimal variety of the conditions required in order to reveal the NLCode's main chromatic features.

The complete photographic sample of the NLCode consists of three sets of photographs, each depicting a different printed NLCode, created and captured under six different configurations. The nonlinear patterns of the three NLCodes were all generated by the Lorenz system (Sec. 3.3, Eq. 3.45). The RK4 method presented in Sec. 2.1, provided a numerical solution of the system for three different sets of ICs, with a total number of iterations $N = 8,000$ for the first and second, and $N = 6,000$ for the third NLCode. Unfortunately, the creation and processing of the NLCode's test photographic sample preceded the parametric study which led to the determination of the scheme's OW (Sec. 4.4). For this reason, the print size S_{Print}^{NLC} of the NLCodes, the thickness factor ℓ_{TH} of the nonlinear patterns, and the working distances W utilised in the creation of the sample photographs, were all selected based on trial and error, and therefore do not closely adhere to the relative specifications the NLCode's OW would dictate for these quantities. However, since the use of the photographic sample focuses solely on the detection of colours without taking any of the spatial attributes of the feature into consideration, and more importantly, since it was seen to successfully produce the anticipated results, the recreation of the following analysis using a new photographic sample was not deemed necessary. All NLCodes have the same print size $S_{Print}^{NLC} = 2.7 \text{ cm}$, which is less than the absolute minimum print size $S_{Print,min}^{NLC,Abs} \simeq 4.44 \text{ cm}$ derived and calculated in Sec. 4.4 (Eq. 4.33). For the first NLCode $\ell_{TH} = 5 \times 10^{-3}$, which is greater than the absolute minimum value 10^{-3} given by Eq. 4.31, and the NLCodes of the other two sets of photographs utilised the maximum estimated value

$\ell_{TH,max}^{Abs}$ of the thickness factor given by Eq. 4.37, and let $\ell_{TH} = 10^{-2}$. All sample photographs were taken at a working distance $W \simeq 6$ cm between the camera and the NLCodes. Lastly, all three NLCodes made use of the same colour theme, defined by the colour base \mathcal{B}^{Hue} (Sec. 3.4, Eq. 3.67), with period $P = 291$ (Eq. 3.64).



Figure 5.4: NLCode’s Test Photographic Sample 1: From left to right, the photographs of the NLCode were taken under 1) daylight, 2) daylight combined with tungsten lighting, 3) a combination of daylight, tungsten lighting and flash, 4) same as (3), 5) tungsten lighting, and 6) same as (5) but with the use of flash. In all photographs except the fourth, the camera is pointing at the NLCode from an intended zero alpha angle. In the fourth photograph $\alpha > 45^\circ$. The rest of the sample specifications are given in the main text.



Figure 5.5: NLCode’s Test Photographic Sample 2: From left to right, the photographs of the NLCode were taken under 1) afternoon light, 2) afternoon combined with tungsten lighting, 3) a combination of afternoon light, tungsten lighting and flash, 4) same as (3) but without the use of flash, 5) tungsten lighting, and 6) same as (5) but with flash. The alpha angles are as described in Fig. 5.4.



Figure 5.6: NLCode’s Test Photographic Sample 3: From left to right, the photographs of the NLCode were taken under 1) daylight, 2) daylight combined with tungsten lighting, 3) a combination of daylight, tungsten lighting and flash, 4) same as (3) but without the use of flash, 5) tungsten lighting, and 6) same as (5) but with flash.

The printer used in this experiment is a consumer model made by Hewlett Packard. The print resolution was set to 300 *ppi*, according to the standards most commercial printing processes comply with. The type of substrate used is a 10 × 15 *cm* glossy photographic paper, chosen as a high quality substrate that minimises ink bleeding and colour blending.

Five of the photographs in each of the three samples were taken under different lighting conditions and from an *alpha angle* (the angle α between the optical axis of the camera and the normal to the NLCode's surface), of approximately *zero degrees*. The lighting conditions tested include *indoors daylight*, *afternoon light*, *tungsten light*, *flash light*, and a few combinations of those. The use of flash in previous tests was seen to significantly improve both the clarity of the image and the vibrance of the colours. Its use in this test sample aims to determine whether there are factors definitively prohibiting its incorporation into the scheme. Lastly, the sixth photograph of each sample was taken from an *alpha angle* greater than 45°, in order to obtain a first, qualitative estimation of the effect α has on the scheme; the distortions introduced due to the perspective projection inherently applied during the capturing of the images, are expected to significantly affect the scheme's performance even for slight deviations of α from 0°. This issue is appropriately treated at a later stage of the scheme's development ([Subsec. 5.2.4](#)). [Figures 5.4 – 5.6](#) present the test photographic sample of the NLCode.

5.2.2.2 Colour Filters – General Chromatic Trends & Background Removal

The ultimate goal of the study presented in this section, is to facilitate the distinction between image pixels that belong to the background of the NLCode's photographs, and pixels of the circular frame and the nonlinear pattern of the NLCode. The image processing performed for this purpose led to the development of two filters enabling the unambiguous identification of background pixels in the NLCode's images. These two filters will be used in the development of the scheme's output palette presented in the next subsection, but also have the potential to further assist the reading process in performing some of its secondary tasks relating to colour detection.

Apple's iPhone 6S Plus outputs its images in a particular RGB colour space, or colour profile, known as standard RGB, or sRGB. The precise specifications of this colour space are provided by the [International Electrotechnical Commission \(IEC\)](#), and its complete name is *sRGB IEC61966-2.1:1999* (IEC, 1999). Depending on the software used to perform the processing of the NLCode's images and in order to ensure that the color vectors referenced and used throughout this study are defined with respect to the correct colour space, there may be a need to convert between the software's default colour space and Apple's device-specific colour profile. Once this is taken care of, the next step is to crop the sample photographs to a suitable rectangle containing the NLCode. This is necessary since reducing the size of the images significantly improves processing time. [Figure 5.7](#) shows all three samples presented in [Par. 5.2.2.1](#), after the cropping process.



Figure 5.7: Test photographic sample after cropping the images using a rectangle defined by its top left vertex and its width and height, in pixel coordinates. From top to bottom, each row corresponds to one of the samples shown in [Figs. 5.4 – 5.6](#).

The first characteristic feature of the NLCode's test photographic sample is observed in the brightness channel of the images, after being converted to the HSB colour space. HSB stands for *Hue*, *Saturation* and *Brightness*, i.e. the three quantities used by the HSB colour space to provide a representation of colours, analogous to the amount of Red, Green and Blue used by the RGB colour space.

Roughly speaking, with hue representing the colour of a pixel based on its perceived similarity with any of the colours in the colour wheel (see e.g. Fairchild, 2013), saturation the colour's purity or richness, and brightness the colour's intensity, the HSB model is said to provide a more intuitive representation of colours than RGB, due to its closest proximity to the way the human eye perceives colours. Figure 5.8 shows the *brightness channels* of the sample images after their conversion to the HSB colour space and the separation of their components.

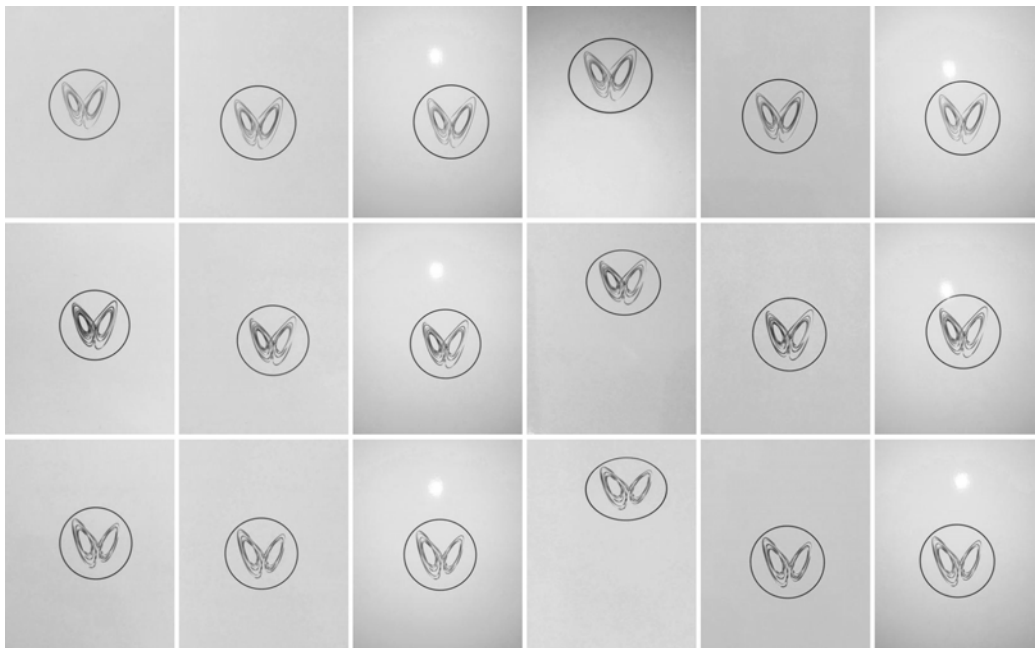


Figure 5.8: Test photographic sample after conversion from RGB to HSB colour space, and decomposition of the sample images to its three components. From top to bottom, each row shows the brightness channels of the cropped samples of Fig. 5.7.

If one considers (a) the circular frame, (b) the nonlinear pattern, (c) the background of the image, and (d) the bright spot created by the camera's flash, as four distinct areas in any photograph of the NLCode, then the clear and unambiguous distinction between these areas during the reading process is a vital part of that process. The brightness channels of the sample images contain information with the potential to enable this distinction. Specifically, the smoothed density function of the pixel intensities in the brightness channel of each sample image, reveals a characteristic chromatic trend followed by the entire sample. Figure 5.9 shows the smoothed density functions of all of the NLCode's images in superposition.

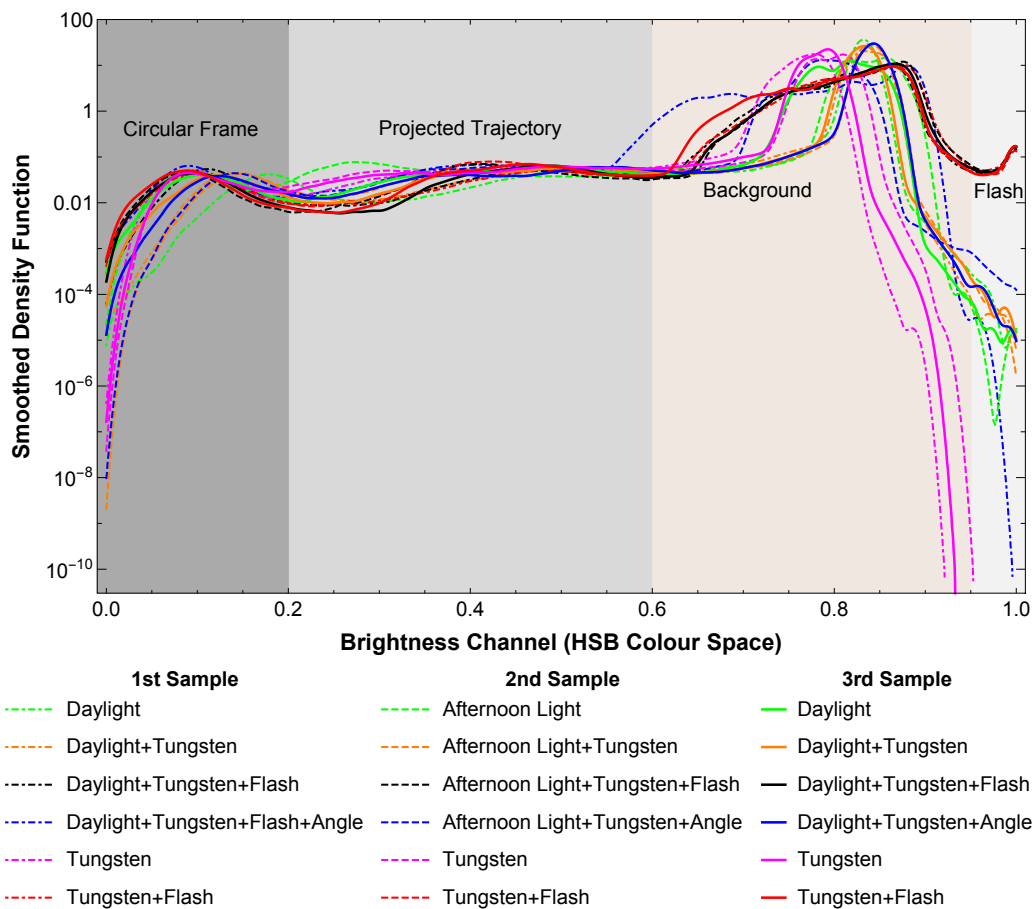


Figure 5.9: Smoothed density functions of the cropped sample images' brightness channels (Fig. 5.8). All curves exhibit the same trend, i.e. local maxima centred around the same approximate brightness intensities, while the curves corresponding to images created using the camera's flash present an additional peak near the maximum brightness value 1. The shaded areas indicate the approximate brightness intervals corresponding to the four distinct areas identified on any photograph of the NLCODE.

The smoothed density functions that correspond to sample images created *without* the use of flash, exhibit – at least in a loose sense of the term – three local maxima located at the approximate brightness intensities 0.1, 0.4 and 0.8. The density functions of the images created using the camera's flash follow the same trend, but also present an additional peak near the highest brightness value 1. Note that the first sample's 4th image – shown in a blue dashed-dotted line – slightly deviates from that trend, because the cropping process removed the flash's bright spot. These observations appear to suggest that the local minima

around the peaks of the density functions may be able to define the brightness intervals characterising each of the distinct image areas previously identified. This brightness-based criterion is summarised in [Table 5.1](#), and is the first filter, called *brightness filter* and denoted by F_{HSB} , applied to the NLCode's sample images.

Table 5.1: The brightness filter F_{HSB} applied to the test photographic sample as a first attempt to separate the four distinct image areas of the NLCode. The superscripts C, P, B and F in F_{HSB} (top row) are used to denote the circle, pattern, background, and flash components of the filter respectively.

| F_{HSB} | F_{HSB}^C | F_{HSB}^P | F_{HSB}^B | F_{HSB}^F |
|----------------------------|-------------|-------------|-------------|-------------|
| Brightness Interval | 0 – 0.2 | 0.2 – 0.6 | 0.6 – 0.95 | 0.95 – 1 |
| Image Area | Frame | Pattern | Background | Flash |
| Colour | Blue | Magenta | Yellow | Black |

The above hypothesis was put to test by first creating a blank image (canvas) of the same dimensions as the cropped images, and subsequently giving each of its pixels a different, distinct colour, depending on the brightness interval its corresponding pixel in the brightness channel belongs to. The results of this test are visually demonstrated in [Fig. 5.10](#). With the exception of the first sample's 4th photograph, the brightness filter seems to confirm the hypothesis made about the brightness channel containing information enabling the distinction between the four image areas of the NLCode. However, certain colour overlaps, i.e. instances of pixels belonging to the NLCode's frame (blue) being identified as pixels of the nonlinear pattern (magenta) and vice versa, as well as pixels clearly belonging to the pattern being identified as background pixels (yellow), suggest that the brightness filter comes with certain drawbacks. The colour overlaps observed are probably due to the brightness limits separating the area of the frame from the area of the pattern and the latter from the background, being inherently ill defined.

The inadequacy of the brightness filter in making a clear distinction between the four image areas of the NLCode, established the fact that any results obtained exclusively from this filter, must be considered unreliable. Its application does however require less logical operations, and for this reason, it will serve as an auxiliary criterion to other tasks (see e.g. [Fig. 5.13](#)). It should also be noted that

from this point onward, the first sample's 4th photograph will be referred to as the "outlier" of the sample, since photographs that use the flash but do not contain the bright spot it creates, are a special case that should be addressed and treated separately, in later stages of the NLCode's development.

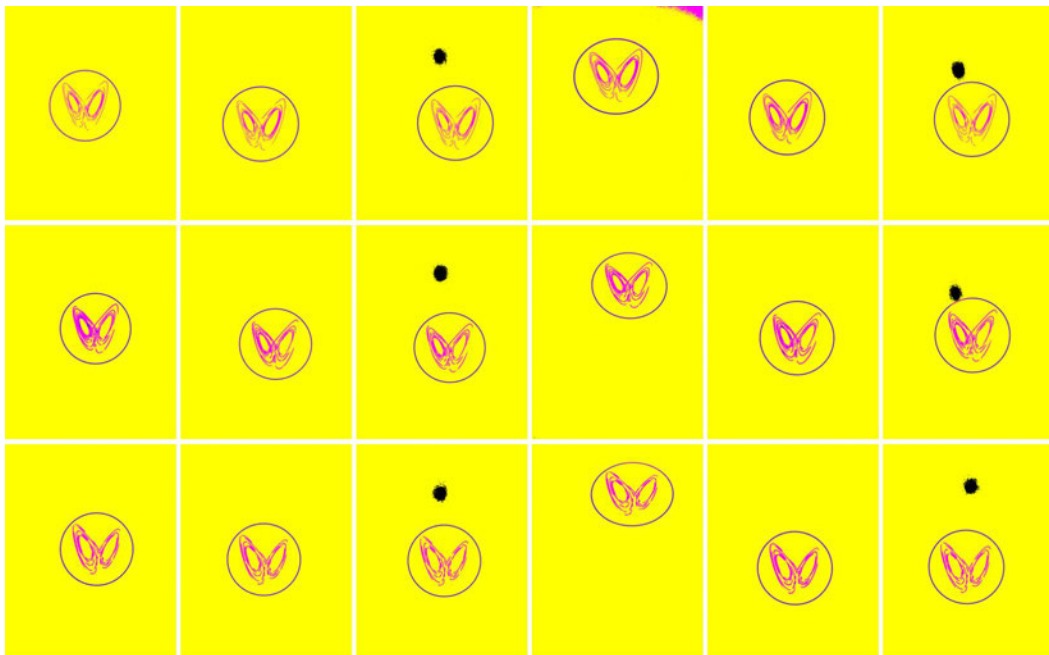


Figure 5.10: Four-colour canvases created by applying the brightness filter F_{HSB} presented in [Table 5.1](#). From top to bottom, each row corresponds to each of the samples shown in [Fig. 5.7](#). A criterion solely based on the brightness of the sample images appears to perform fairly well in identifying pixels that belong to the image background and the bright spot created by the camera's flash. However, the spatial mixing of blue and magenta pixels, as well as the pixels of the nonlinear pattern being incorrectly identified as background pixels (yellow), highlight F_{HSB} 's shortcomings.

The next step of this study moves toward the development of a tool enabling the unambiguous identification of background pixels, including the flash's bright spot. The images shown in [Fig. 5.7](#) were processed in order to identify sixteen of the most dominant colours they contain. After that process was complete, the dominant colours most likely corresponding to either the frame or the pattern, were removed. The list of colours this process led to is shown in figure [Fig. 5.11](#).

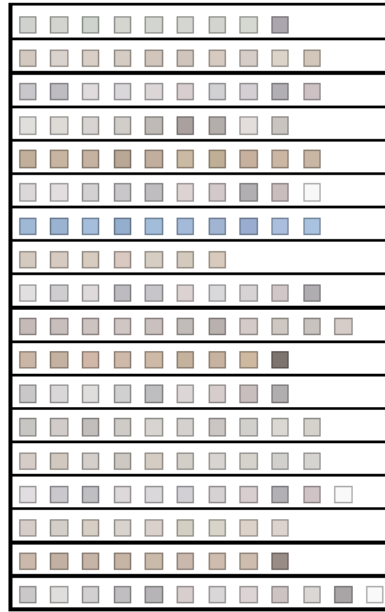


Figure 5.11: List of the dominant colours found in the background of the NLCode's sample images. From top to bottom, each row corresponds to one of the sample photographs shown in figure Fig. 5.7. Note the distinctive light blue colour in the list (7th row) corresponding to the photograph taken in the afternoon, with no other identified light sources present.

The most efficient way found to utilise the colours shown in figure Fig. 5.11 in order to identify background pixels, is through the range of colours they cover in the RGB colour space. This range forms the second filter developed in this section, and is defined as

$$F_{RGB} = \{\{r_{min}, r_{max}\}, \{g_{min}, g_{max}\}, \{b_{min}, b_{max}\}\}, \quad (5.11)$$

where r_{min} and r_{max} are the minimum and maximum intensities in the red channels of all the colours shown in Fig. 5.11, and g and b denote the respective quantities in the green and blue channels. The precise values found are given below.

$$F_{RGB} = \{\{0.494, 0.981\}, \{0.418, 0.975\}, \{0.436, 0.977\}\}. \quad (5.12)$$

Note that depending on the software used for image processing, the colour intensities in each channel may be given in the range $[0.0, 1.0]$, as is the case here, or in the range $[0, 255]$. The conversion from one representation to the other, should such need arise, is as simple as multiplying an intensity in $[0.0, 1.0]$ by 255 and rounding to the nearest integer.

In order to test its efficiency in identifying background pixels, F_{RGB} was applied to the sample images shown in figure Fig. 5.7. Starting again with a blank canvas of the same dimensions as the processed images, every pixel whose colour intensities in the r , g and b channels belonged to the corresponding intervals provided by Eq. 5.12 were coloured yellow, while the rest of the pixels were coloured blue. Figure 5.12 shows the results of this test. The filter was quite successful in removing all background pixels, except those belonging to the flash's bright spot in the relevant images. However, in the case of the first sample, it incorrectly identified pixels belonging to the NLCode's pattern, as being background pixels. This observation simply stresses the importance of the theoretically derived tolerances of the parameters shaping the scheme's OW (Sec. 4.4).

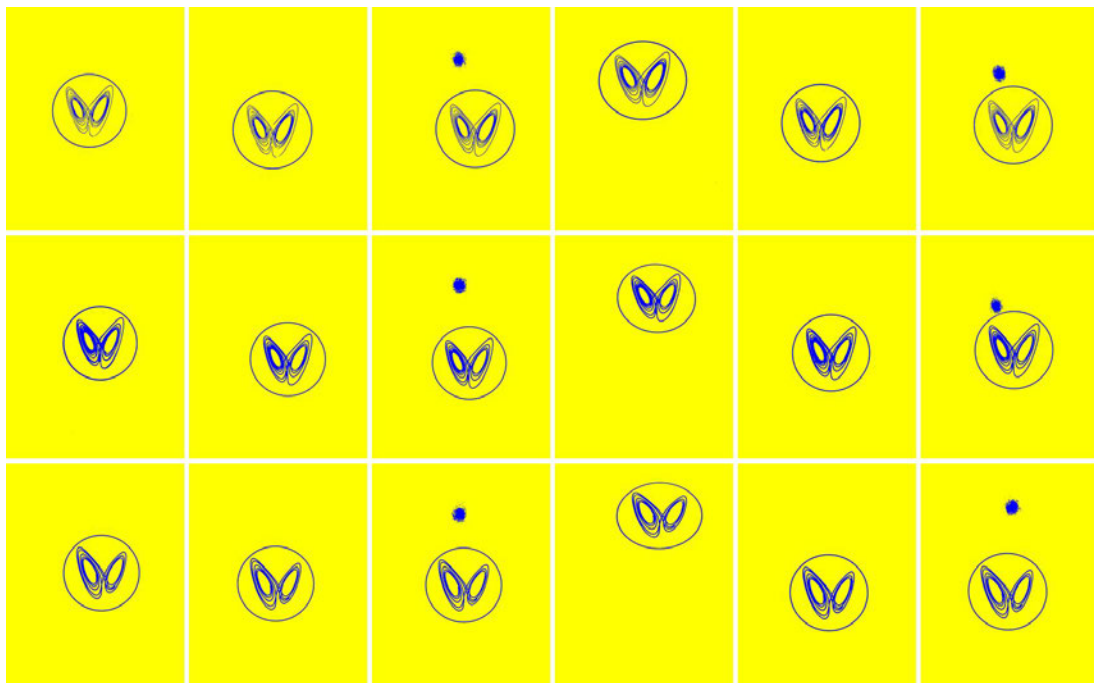


Figure 5.12: Two-colour canvases created using the filter F_{RGB} developed for background removal (Eq. 5.12). From top to bottom, each row corresponds to each of the samples shown in Fig. 5.7. Every pixel identified as a background pixel is coloured yellow, and the rest of the pixels are shown in blue. The filter was partly successful, but failed to remove the pixels of the flash's bright spot. In the case of the first sample (top row), it also incorrectly identified pixels belonging to the nonlinear pattern, as background pixels.

The issue concerning the bright spot created by the camera's flash was successfully treated by incorporating to the pixel classification process the brightness filter F_{HSB} previously developed. In this final test, for a pixel to be identified as a background pixel, its RGB values should belong to the intervals of F_{RGB} given by Eq. 5.12 – as in the previous test, *and/or* its brightness intensity to the interval $F_{HSB}^F = [0.95, 1]$ corresponding to the flash area of the filter defined in Table 5.1. The results are shown in Fig. 5.13.



Figure 5.13: Two-colour canvases created using the F_{RGB} filter developed for background removal (Eq. 5.12), combined with the F_{HSB}^F component of the brightness filter corresponding to the flash area (Table 5.1). The ordering of the samples presented and the colour scheme used is the same as in Fig. 5.12. The combination of these two filters managed to successfully identify every background pixel in the images, including the pixels of the flash's bright spot.

5.2.2.3 Colour Filters – Output Palette

The colour profile of the NLCode was introduced in Sec. 3.4, where it was defined as a *periodic sequence* \mathcal{C} of colours that form a *smooth gradient* (Eq. 3.62). Sequence \mathcal{C} is defined by a *colour base* \mathcal{B} of elementary colours (Eq. 3.63), a periodic (sawtooth) function $f(n; P)$ (Eq. 3.64) that makes \mathcal{C} periodic with period

P , and a linear interpolation formula (Eq. 3.66) that creates the desired gradient effect. In Sec. 4.3, where the NLCode Generator deals with COs, the colour base \mathcal{B} was defined as the *input palette* \mathcal{P}_{in} of the colour profile (Eq. 4.17), used for colour comparison in the detection of COs. The present paragraph completes the discussion around the scheme's colour profile, with the development of its last basic component, called the *output palette* and denoted by \mathcal{P}_{out} .

At the beginning of Subsec. 5.2.2, it was mentioned that the output palette consists of a certain number of colours a camera *detects* on the line representing the NLCode's pattern. A simple identification of any number of detected colours, however, each of which can – and must – be associated with different groups of pixels on the nonlinear pattern, creates a colour palette which lacks the structure required in order to assist the reading process in a systematic way. For this reason, the output colours need to be selected in such a way, so as to meet the two following conditions:

- Each output colour must associate with pixels of the nonlinear pattern that form *simply connected segments* on the line representing it. In other words, when the detected colours of the pixels on the pattern are matched – according to some rule – to the colours of the output palette, the pattern must consist of a sequence of line segments, each containing pixels of a *single* output colour.
- The sequence of output colours assigned to the line segments the pattern consists of – after the processes of colour detection and matching have been completed – must be *periodic*. This means that the colours in the output palette should form a sequence, i.e. they should be ordered, and that sequence must repeat itself on the pattern in a canonical fashion.

The development of an output palette that satisfies both of the above conditions, is an iterative process which takes the following steps:

1. The first step prepares the sample images of the NLCode shown in Fig. 5.7 for processing, by removing their background using the two filters F_{RGB} (Eq. 5.12) and F_{HSB}^F (Table 5.1) developed in the previous paragraph. Figure 5.14 shows the NLCode's cropped sample with its background removed.



Figure 5.14: The NLCode's cropped sample with the image background removed, using a combination of the filters F_{RGB} and F_{HSB}^F developed in [Par. 5.2.2.2](#). From top to bottom, each row corresponds to each of the cropped samples shown in [Fig. 5.7](#). The process started with blank canvases of the same dimensions as the cropped images. The pixels identified as background pixels were left white, and the rest of the pixels were given the colour of their corresponding pixels in the original cropped images.

2. The periodic sequence \mathcal{C} representing the NLCode's colour profile ([Eq. 3.62](#)) consists of N colours, where N is the length of the NLCode's 2D projected trajectory. This step's task is to select a number N_{out} of these colours, and without permuting them, form a sequence with sufficient contrast between adjacent colours. Note that N_{out} is expected to be of the order of N_{in} , which is the number of non-repeating colours in the input palette (see [Eq. 3.63](#)).
3. This next step takes into consideration the fact that the captured colours on the NLCode's pattern, are dark shades of the input colours sequence \mathcal{C} consists of (also see [Fig. 5.17](#)). The first, temporary version of the output palette \mathcal{P}_{out} created in light of this observation, consists of dark shades of the N_{out} colours selected in step 1, with the addition of black, which is the colour of the circular frame. Note that the vague use of the term "dark shade" is not an overlook. Depending on the software used for the processing of the sample images, the "darkness" of a colour may be quantified in different ways. However, a detailed account of the rules followed by built-in functions performing such tasks, far exceeds the purposes of this study.

4. Assuming the vector representation of colours introduced in [Sec. 3.4](#), the colour of each pixel in the NLCode's pre-processed sample images ([Fig. 5.14](#)), has a certain (euclidean) distance from each of the colours in the temporary output palette (also see [Sec. 4.3](#), [Eq. 4.16](#)). This step creates $N_{out} + 1$ lists, each associated with one of the temporary output colours, by classifying the detected colours according to the following criterion: If the distance between the colour of a pixel and a temporary output colour is less than a predefined threshold distance d_{crit} , that colour enters the list associated with the output colour it was tested against. Note that the above calculations will only be consistent if the temporary output colours defined in step 3 refer to the same colour space as the pixel intensities drawn from the preprocessed images.
5. This step creates a new temporary output palette, by letting each of the first N_{out} lists created in the previous step, be represented by the *mean* of the colours it contains. Each of these "mean" detected colours is denoted by \mathbf{d}_i , $i = 1, \dots, N_{out}$. The list associated with the black output colour is processed the same way as the background colours were processed in the previous paragraph ([Par. 5.2.2.2](#)). This means that the last list is not represented by a "mean" colour, but by a *range* of colours, denoted as F_K (specified in [Eq. 5.14](#)), that is given the form of the background filter F_{RGB} defined by [Eq. 5.11](#). Therefore, this temporary output palette is a union of the general form

$$\mathcal{P}_{out} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_{N_{out}}\} \cup F_K. \quad (5.13)$$

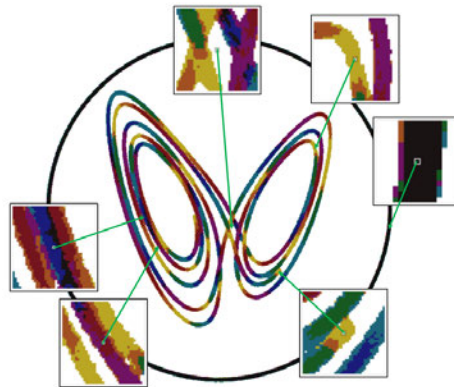
6. The new temporary output palette is applied to the NLCode's preprocessed sample images according to the following rules, applied in succession:
 - Each pixel of a blank canvas is given the mean output colour \mathbf{d}_i that is closest, in the sense of the Euclidean distance introduced in step 4, to the actual colour detected in the corresponding pixel of the preprocessed sample image.
 - Each pixel already coloured according to the previous rule, is subjected to a second test. If the intensities of all three channels of the pixel's original colour belong to the r , b , and g ranges of F_K , then that pixel is coloured black.
7. The sample images processed in the previous step are evaluated for compliance with the two conditions imposed on the output palette. If the results are satisfactory, then the second temporary output palette created in step 5 becomes the final output palette of the NLCode's colour profile. If the segments the non-

linear pattern consists of are not *uniformly* coloured to a sufficient degree, but instead each presents a mixture of colours, then the process is repeated from step 2 with a different selection of input colours and/or different dark shades of the input colours selected.

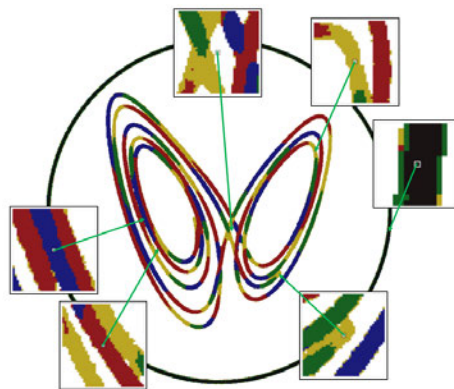
Figure 5.15 demonstrates the development process described above with two examples, both of which make use of the first image of the NLCode depicted in the third sample (Fig. 5.7). The output palette applied in the second example of Fig. 5.15 is the final version of the palette developed by the iterative process presented in this paragraph. According to its general form given by Eq. 5.13, \mathcal{P}_{out}^{Hue} is the union of the sequence comprised by four mean detected colours and the black filter F_K , both computed in step 5. The numerical representation of the output palette with respect to the RGB colour space, is the following:

$$\mathcal{P}_{out}^{Hue} = \underbrace{\{(0.73, 0.66, 0.15), (0.57, 0.11, 0.12), (0.11, 0.11, 0.48), (0.09, 0.38, 0.13)\}}_{\text{mean output colours } \mathbf{d}_i, i = 1, \dots, N_{out} = 4} \cup \underbrace{\{\{0.00, 0.17\}, \{0.00, 0.17\}, \{0.00, 0.17\}\}}_{r, g, b \text{ ranges of the black filter } F_K}. \quad (5.14)$$

Figure 5.15a illustrates a case in which the temporary output palette created in step 5 failed to satisfy both conditions imposed on it. In this example, the number N_{out} of colours selected in step 2 is 7, namely yellow, orange, red, magenta, blue, cyan, and green, i.e. $N_{out} > N_{in}$. Due to an insufficient contrast between adjacent colours in the temporary output palette created in steps 2 – 5, the line segments of the nonlinear pattern are not uniformly coloured. Apart from the bleeding of adjacent colours into one another's segments, the presence of orange in the temporary output palette allowed this colour to appear in several transition areas from green to yellow, which interfered with the ordering of the colours in a non-reliable manner. These observations are clearly seen in the zoomed-in areas superimposed on the processed image. Note also, that in some of these zoomed-in areas, pixels of the pattern are coloured black, while various colours of the output palette appear in the area of the NLCode's circular frame. This is due to an earlier version of step 5, in which all the $N_{out} + 1$ lists created in step 4 were represented by the mean of the colours they contained, including black, and a version of step 6 which only included the first rule for colour matching.



(a) The output palette applied in this example includes dark shades of the seven colours yellow, orange, red, magenta, blue, cyan, and green, and made use of the earlier versions of steps 5 and 6 previously described. This output palette failed the first condition imposed on it by providing non-uniformly coloured line segments, and the second by interleaving orange segments in the transitional areas between green and yellow, in a non-reproducible manner, which cannot be reliably considered systematic.



(b) The output palette in this example consists of dark shades of the four colours yellow, red, blue, and green. Its development made use of the updated versions of steps 5 and 6 presented in this paragraph, and led to the fulfillment of both conditions, by producing uniformly coloured line segments that repeat along the projected trajectory in a periodic fashion.

Figure 5.15: Visual demonstration of the process followed for the development of the NLCode's output palette. Both figures show the first image of the third test sample of the NLCode (Fig. 5.7, bottom left). Figure 5.15a illustrates a failed attempt to create an output palette satisfying the two conditions of uniformity and periodicity described in the beginning of this paragraph. Figure 5.15b shows the result of the final, successful output palette created by the iterative process of development. Both figures include snapshots of zoomed-in areas superimposed on the processed images, which illustrate the details discussed in the main text.

Figure 5.15b presents a significantly improved picture. In this case, the colours selected in step 2 were reduced to $N_{out} = 4$, namely yellow, red, blue, and green. The line segments of the pattern are *uniformly coloured*, with minor imperfections which are not expected to affect the scheme’s performance. The significant reduction of the size of the output palette – from 6 non-repeating colours (the same as the input palette) to 4 in the final output palette – also appears to have allowed the second condition imposed, i.e. that of *periodicity*, to be fulfilled. Finally, the updated version of step 5, which identifies black pixels by range, and of step 6 which includes the second rule for colour matching, prevented pixels of the pattern from being coloured black, and enabled the correct identification of the vast majority of pixels comprising the NLCode’s circular frame. However, the circular frame still appears to have several of its pixels coloured by mean output colours. This is actually what motivated the spatial separation of the two main elements of the NLCode, using the masks presented in Subsec. 5.2.1 (Fig. 5.3).

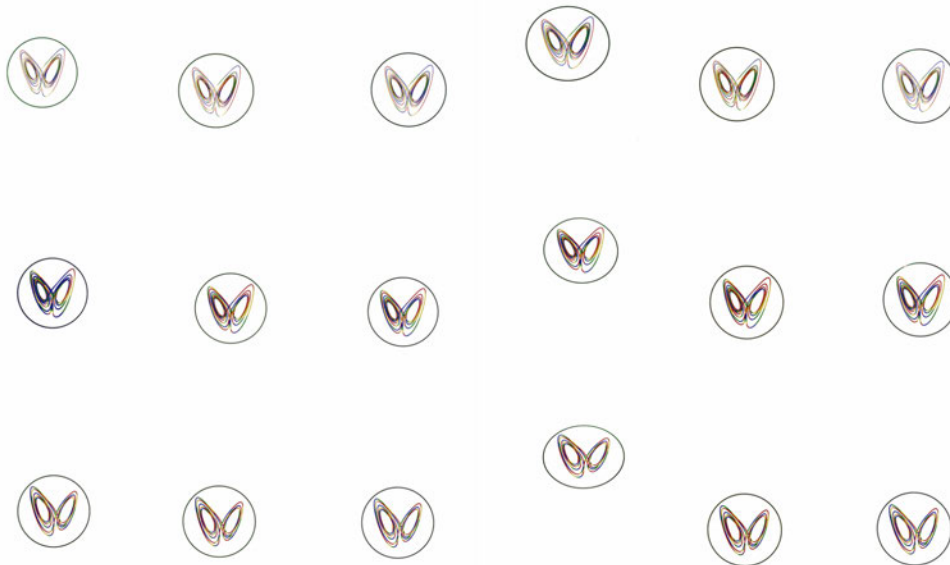


Figure 5.16: The NLCode’s cropped sample, after background removal (Fig. 5.14) and the subsequent application of the output palette \mathcal{P}_{out}^{Hue} developed in this paragraph (Eq. 5.14). From top to bottom, each row corresponds to each of the cropped samples shown in Fig. 5.7. The process started with blank canvases of the same dimensions as the cropped images. The white background pixels were ignored, and the remaining pixels belonging to either the NLCode’s pattern or its circular frame, were coloured according to the rules described in step 6 of the development process. The palette managed to meet both conditions imposed on it, by producing uniformly coloured line segments that repeat on the nonlinear pattern in a periodic fashion.

Figure 5.16 shows the NLCode's preprocessed images (Fig. 5.14), after the application of the final output palette \mathcal{P}_{out}^{Hue} to the entire sample. Zooming in on any of the sample images gives a picture similar to the detailed picture provided in Fig. 5.15b, proving that the output palette has performed equally well across the sample.

Figure 5.17 summarises the NLCode's colour profile and highlights some important properties of its main components. The input and the output palettes of the profile are shown as lists of colours on each side of the figure. Apart from the colours in each of them, these palettes exhibit two more important differences. The first difference was already mentioned in the beginning of the present subsection (Subsec. 5.2.2), and has to do with the fact that while the input palette \mathcal{P}_{in}^{Hue} is a predefined feature of the NLCode, the output palette \mathcal{P}_{out}^{Hue} needs to be developed through the processing of the NLCode's sample images. Secondly, while the input palette has $N_{in} + 1$ components and the output palette $N_{out} + 1$, (a) since the output palette draws its original colours from the NLCode's colour sequence \mathcal{C} (step 2), N_{out} is generally different than N_{in} , and (b) the "plus 1" component in each palette is different and serves an entirely different purpose. In the case of \mathcal{P}_{in}^{Hue} , the additional component is the first colour of the palette, repeated at the end of the sequence for purposes that have been thoroughly explained in Sec. 3.4. The output palette has no need for this kind of repetition. The last component of \mathcal{P}_{out}^{Hue} is the black filter F_K incorporated to the palette in order to identify as many pixels of the NLCode's circular frame as possible.

The circular segments overlaid on the colour wheel at the center of Fig. 5.17 give a visual sense of the range of input colours covered by each of the output colours. The overlapping of these segments implies that pixels with orange, magenta, cyan, and green-yellow hues, can be classified as either of the two overlapping output colours covering their hue range, depending on their other two colour coordinates, i.e. saturation and brightness. This explains the bleeding, albeit uniform, of adjacent colours into one another's line segments observed even in Fig. 5.15b. Finally, the areas of the circular segments roughly correspond to the relative percentages of pixels belonging to the nonlinear pattern, associated with each of the output colours. The exact percentages involved can easily be calculated and even help in the development of the output palette. However, such an implementation will only be considered if the use of colours indeed proves to be valuable to the reading process.

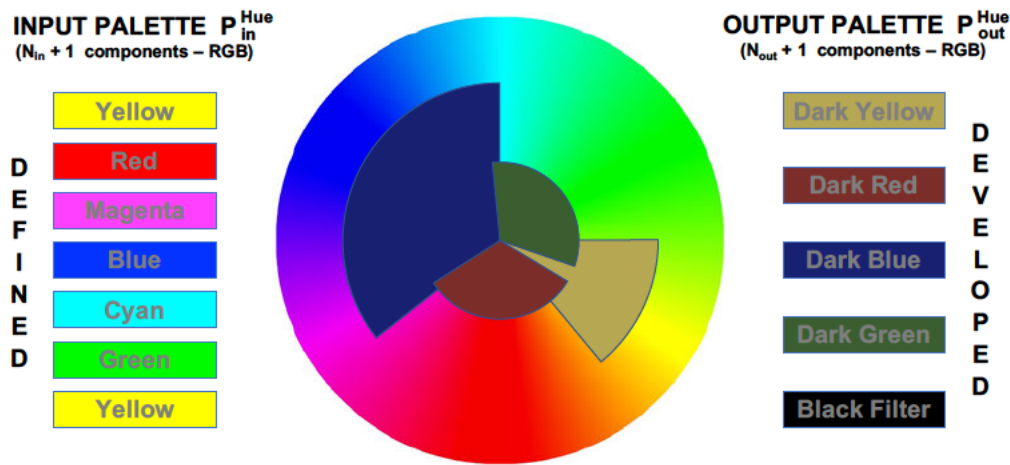


Figure 5.17: Basic components of the NLCode's colour profile. The input palette \mathcal{P}_{in}^{Hue} (Sec. 3.4) and the output palette \mathcal{P}_{out}^{Hue} (Eq. 5.14) are shown to the left and right sides of the figure respectively. The colour wheel in the middle, is overlaid with circular segments indicating the range of input colours covered by each of the output colours. The overlapping of these segments is due to the fact that this illustration only considers one of the three colour coordinates, namely the hue, while ignoring the saturation and brightness required to uniquely identify a colour.

The computational intensity of the image processing performed in the last two sections may raise a concern regarding the scheme's performance. Regardless of whether the aim was the identification of pixels belonging to the background of the sample images or the NLCode's pattern, each process implemented scanned the entire sample, performing logical or numerical operations on a pixel-by-pixel basis. The important point to clarify, is that these processes were implemented strictly for the development of the filters related to background removal and the output palette. Even though the computational power required by these processes can be harnessed, albeit not very easily, from a smartphone, the reading process has no need to scan an entire image before it reaches a given point of interest. Instead, it is designed in a way that allows it to navigate an image mostly based on geometrical considerations, using the pixel coordinate system of the NLCode's captured image. Once a given pixel on the projected trajectory is located, the main process responsible for tracing the pattern takes over, and the background filters and output palette are used as auxiliary guides to that main process.

5.2.2.4 Application of Colour Filters

Figure 5.18 shows the third view of the Demo App. This view is called *masked photo view* because it displays the masked image of the NLCode's arced circular frame (Subsec. 5.2.1, Fig. 5.19, center). The second masked image, i.e. that of the nonlinear pattern (right of Fig. 5.19), is subject to all the necessary image processing techniques in preparation for the reading process, and is only displayed in the Demo App's last views. The UI on the masked photo view presents the user with a few options, each corresponding to a UIE displayed on the view. From this point onward, only UIEs that have not been described in earlier views of the Demo App will be discussed.

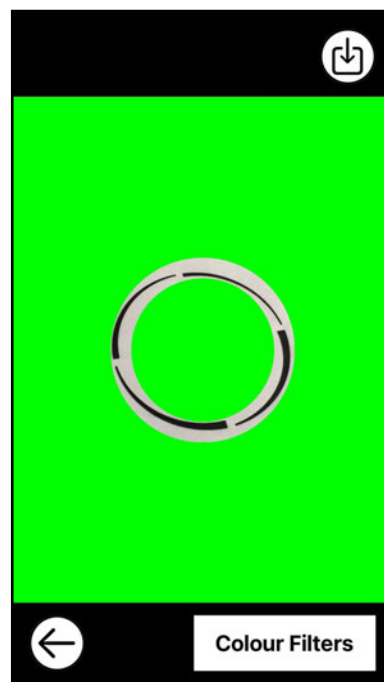


Figure 5.18: Demo App - Masked Photo View & Colour Filter prompt: The third view of the Demo App shows the masked image of the NLCode's arced circular frame (Fig. 5.19, center). The target Area Button is no longer available to the user, but the Save Button still offers them the option to save the masked image to their Photo Library. By tapping the Back Button, the user can still move to the previous views, if they wish to. The Colour Filters Button displayed at the bottom-right of the view, prompts the user to proceed with the processing of the masked image(s). When the Colour Filters Button is tapped, the colour filters developed in Pars. 5.2.2.2 and 5.2.2.3 are applied to the masked images of the separated elements of the feature, accordingly.

- **Colour Filters Button (bottom view, right)** – The main control UIE of the masked photo view is the Colour Filters Button, which enables the user to initiate the application of the *colour filters* developed in [Pars. 5.2.2.2](#) and [5.2.2.3](#). The manner in which those filters are applied to the two masked images created in [Subsec. 5.2.1](#) are the main topic of this paragraph. The Colour Filters Button’s icon simply displays the text *Colour Filters*.

The development of the colour filters presented in [Pars. 5.2.2.2](#) and [5.2.2.3](#), was based on the test photographic sample of the NLCode ([Par. 5.2.2.1](#), [Figs. 5.4 – 5.6](#)), and produced filters that successfully made the distinction between four areas of a captured image of the NLCode (circular frame, pattern, background, and flash spot), based on its chromatic attributes (see [Fig. 5.9](#) and related discussion). However, that analysis had to treat each sample image as a whole, that is, without considering any spatial characteristics that might help make at least some of the necessary distinctions. After the application of the masks created in [Subsec. 5.2.1](#), this is no longer the case.

The masked image separating the arced circular frame from the rest of the NLCode’s image ([Fig. 5.19](#), center), only contains the frame itself, and the background of the image; it is assumed that the flash spot, if at all present, does not fall right on the circumference of the frame, which would cause the corner detection algorithm ([Subsec. 5.2.3](#)) to fail. The colour filters applied on that masked image are the F_{RGB} filter for background removal ([Eq. 5.12](#)), combined with the component of the F_{HSB}^F brightness filter corresponding to the flash spot ([Table 5.1](#)). The result of this combined filter on the masked image of the frame, is the image on the left of [Fig. 5.19](#). As is clearly obvious – and expected, using the same colour filters, the filtering of the masked image is superior to the filtering attempting to detect the frame using an unmasked image (see e.g. [Fig. 5.15](#)). The arcs of the frame are uniformly coloured black, that is, no colour mixing is present, and well defined.

The masked image only exposing the nonlinear pattern of the NLCode ([Fig. 5.19](#), right), also only contains the pattern itself and the background; again, the flash spot is assumed to only contribute a halo around the feature, without essentially erasing parts of it. The filters applied in this case are the F_{RGB} and the F_{HSB}^F as before, followed by the *mean* component of the output palette \mathcal{P}_{out}^{Hue} ([Eq. 5.14](#)). The black RGB filter F_K was not used in this case, as it is unnecessary, and could compromise the integrity of the dark colours on the

pattern. The result of this filtering process is the image on the right of Fig. 5.19. The pattern is well defined and fairly uniform and the colours of the output palette follow a canonical order and repeat periodically. However, there is some colour mixing present, between the colours of sequential and/or adjacent segments.

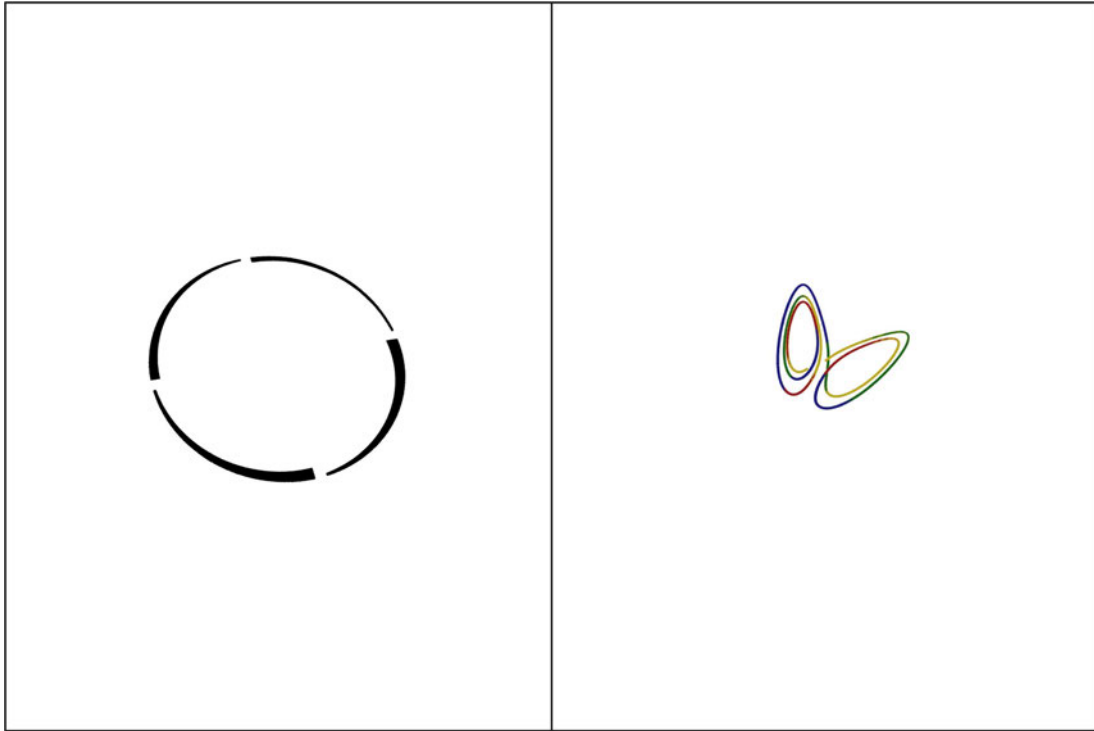


Figure 5.19: Colour detection on the NLCode's masked basic components: The image on the left is the masked image of the frame, after applying a combination of the F_{RGB} filter for background removal, and the component of the F_{HSB}^F brightness filter corresponding to the flash spot. The image on the right is the masked image of the NLCode's pattern, after applying the combination of the same F_{RGB} and F_{HSB}^F filters, followed by the mean component of the output palette \mathcal{P}_{out}^{Hue} . Both filtering processes show an obvious improvement compared to the filtering performed on the unmasked images of the test photographic sample of the NLCode (see e.g. Fig. 5.15). The main features of the NLCode depicted are well defined, and the colours of the output palette follow a canonical order and repeat on the pattern periodically. However, while the arced circular frame is uniformly coloured black, the nonlinear pattern shows some mixing in the transition between colours of its sequential, or adjacent segments.

5.2.3 Corner Detection & Identification

Figure 5.20 shows the fourth view of the Demo App. This view is called *filtered photo view* because it displays the filtered image of the NLCode's arced circular frame (Par. 5.2.2.4, Fig. 5.19, left). The second filtered image, i.e. that of the pattern (right of Fig. 5.19), is only displayed in the Demo App's last views, where it is used in the reading process. The UI on the filtered photo view presents the user with a few options, each corresponding to a UIE displayed on the view.

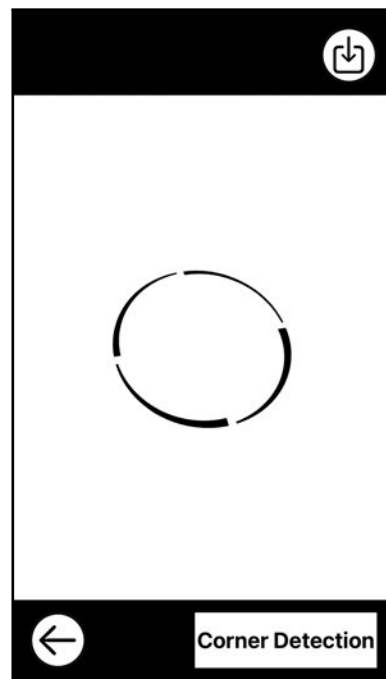


Figure 5.20: Demo App - Filtered Photo View & Corner Detection prompt: The fourth view of the Demo App shows the filtered image of the NLCode's arced circular frame (Fig. 5.19, left). The Corner Detection Button displayed at the bottom-right of the view, prompts the user to proceed with the processing of the displayed image. When the Corner Detection Button is tapped, the Demo App applies on the image a corner detection algorithm, followed by an identification process pairing each corner with the marked position on the arc it belongs to – beginning, outer ending, inner ending – as well as the arc itself – 1st, 2nd, ... n_A -th. The remaining UIEs available to the user are the same as in the previous view of the Demo App (Fig. 5.19).

- **Corner Detection Button (bottom view, right)** – The main control UI of the filtered photo view is the Corner Detection Button, which enables the user to initiate the application of a *colour detection algorithm* on the filtered image of the frame, and the subsequent identification process of the detected corners. The corner detection algorithm employed and the identification process developed are discussed in this subsection.

By design, the corners of the frame's arcs are meant to be used for (a) the correction of the perspective distortions introduced by the perspective transform applied on the image of any scene, when captured by a camera, and (b) the application of the method of least squares adjustment, in order to calculate the matrix used for the transformation between plot and pixel coordinates (Subsec. 3.2.2).

The algorithm used for the detection of the arcs' corners is the *Shi-Tomasi corner detection algorithm* (Shi & Tomasi, 1994), which is very commonly used by many software applications performing tasks related to feature detection on images. The incorporation of the algorithm into the Demo App made use of Brad Larson's implementation, which is included in the open source project for GPU-accelerated image and video processing, called *GPUImage framework*. This framework's second generation – *GPUImage 2* – is written in the Swift programming language, for various platforms, including iOS (Larson, 2019). The Shi-Tomasi corner detector in this implementation pre-processes the image using a *Gaussian blur* (see e.g. Reinhard et al., 2006), in order to smooth out crisp edges or isolated pixels which might mistakenly be identified as corners. The algorithm takes certain arguments as parameters; the Demo App uses the default values of all parameters related to the corner detector, and only adjusts the one parameter of the Gaussian blur. This parameter is called *radius of the Gaussian blur* (specified in pixels), and is related to the standard deviation of the Gaussian distribution used for smoothing; its default value is 2.0, and the Demo App sets it to 4.0 in order to apply a stronger blur on the image.

The image on the left of Fig. 5.21 shows the result of the corner detection process. The detected corners are drawn on the filtered image of the frame, as red crosses. Zooming in on that image, shows that all $3n_A = 12$ corners of the frame have been detected, and with an impressive accuracy. A visual inspection is all the assessment the corner detector can have at this stage. Its ultimate performance is of crucial importance to the scheme, but can only be assessed through the success of the processes dependent on it.

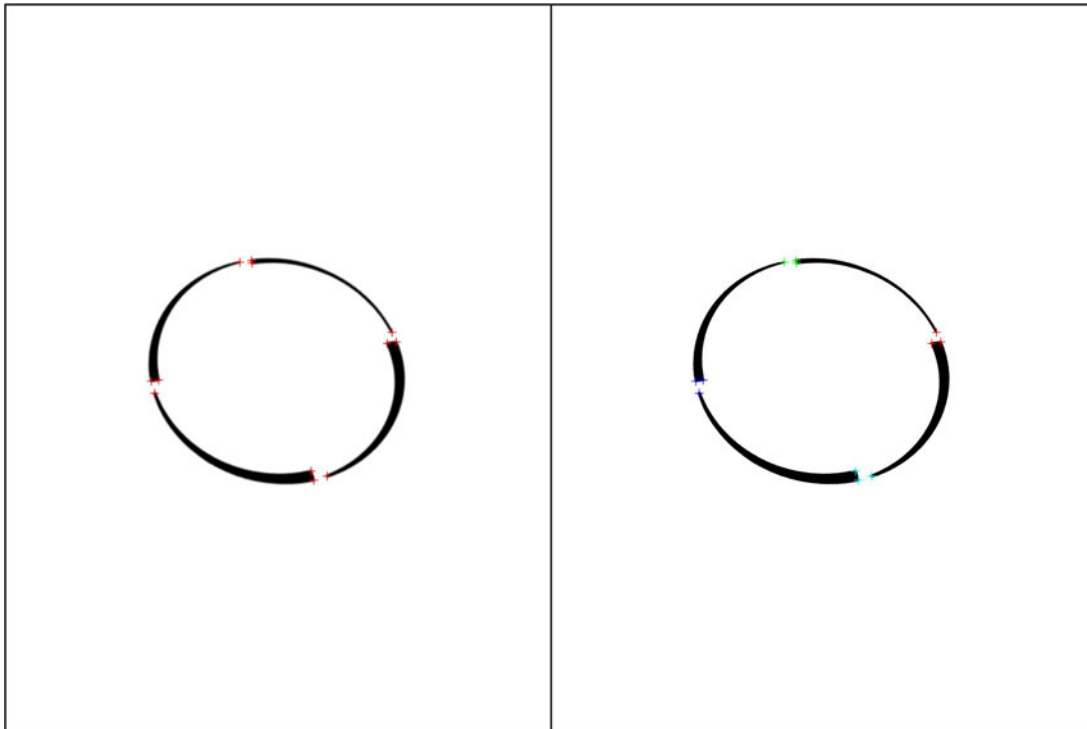


Figure 5.21: Corner detection on the NLCode's arced circular frame: The image on the left is the filtered image of the frame, with the detected corners drawn on it as red crosses. The Gaussian blur-facilitated Shi-Tomasi corner detector, has detected all corners of the arcs with excellent accuracy, but its overall performance can only be evaluated implicitly, through the performance of the processes dependent on it. The image on the right has the detected corners drawn in different colours per triad, indicating the quadrant of the frame each corner triad should be, in the printed NLCode, i.e. before the distortions introduced by capture. The colours red, green, blue, and cyan, are used to indicate corners that should be in the 1st, 2nd, 3rd, and 4th quadrant respectively. The detected corners are also identified in terms of their relative position in the arcs they belong to, i.e. as arcs' beginnings, outer endings and inner endings. However, this distinction is not shown in this image.

The process developed for the identification of the arc each of the detected corners belongs to, as well as its relative position on its respective arc, performs the following steps in succession:

1. The detected corners are first clustered into four groups of three corners each, using a *neighbourhood-based clustering algorithm* that uses the Euclidean distance as a dissimilarity measure (see e.g. G. Gan et al., 2007).

2. The corner triads are placed in ascending order, by the area of the triangle they form. Given the increasing width of the arc endings, and the fact that, in the printed NLCode – before capture, the 1st, 2nd, 3rd, and 4th arc's beginnings are on the 1st, 2nd, 3rd, and 4th quadrant of the frame respectively, the ordered corner triads are matched with the frame quadrants they *should* be in, as follows: The first triad (smallest triangle area) corresponds to the 2nd quadrant, the second triad to the 3rd quadrant, the third triad to the 4th quadrant, and the fourth triad to the 1st quadrant. This matching process is very sensitive to the relative widths of the arcs endings and the size of the gaps between the arcs. Apart from carefully choosing those frame parameters, the result of this matching can be complemented/replaced by a second matching criterion, based on the angles of the corner triads, relative to the horizontal.
3. Using the centroid of the entire set of detected corners as representative of the frame's center, the three corners in each cluster are then identified as arc's beginnings, outer endings, and inner endings, as follows: The angles formed by every pair of corners in a triad with the centroid, are calculated and placed in ascending order. The smallest angle is formed by an arc's outer and inner endings with the centroid; the outer (inner) ending is the corner farthest (closest) to the centroid. The remaining corner of the triad is an arc's beginning.
4. The final output of this process takes into account the fact that the 1st quadrant contains the 1st arc's beginning and the *last* arc's endings, the 2nd quadrant contains the 2nd arc's beginning and the 1st arc's endings, and so on.

The image on the right of [Fig. 5.21](#) shows the result of the corner identification process. In this case, each corner triad is drawn with a different colour, depending on the quadrant its corners *should* be in, that is, in the printed NLCode before capture. The colours red, green, blue, and cyan, correspond to the 1st, 2nd, 3rd, and 4th quadrant respectively. It is stressed that the sensitivity of this identification process does not only depend on the frame specifications, but also on the perspective distortions introduced during capture, which can alter lengths significantly. However, if one triad can be identified reliably – this is the triad with the smallest triangle area – the rest of the corner clusters can be identified using relative angle-based criteria.

5.2.4 Perspective Correction

Figure 5.22 shows the fifth view of the Demo App. This view is called *detected corners view* because it displays the image of the NLCode's arced circular frame, with the detected corners – colour-coded with respect to the frame quadrant assigned to them – drawn on it (Subsec. 5.2.3, Fig. 5.21, right). The UI on the detected corners view presents the user with a few options, each corresponding to a UIE displayed on the view.

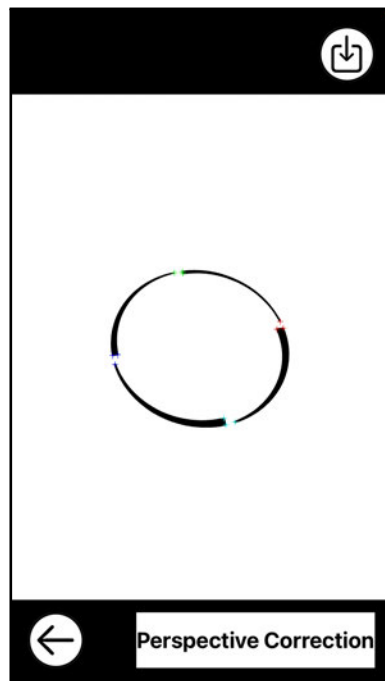


Figure 5.22: Demo App - Detected Corners View & Perspective Correction prompt: The fifth view of the Demo App shows the filtered image of the NLCode's arced circular frame (Fig. 5.21, right), overlaid with the corners of its arcs, as they were detected and identified in Subsec. 5.2.3. The Perspective Correction Button displayed at the bottom-right of the view, prompts the user to proceed with the processing of the displayed image. When the Perspective Correction Button is tapped, the Demo App applies on the image a perspective transform, in order to correct the perspective distortions introduced by the capture of the NLCode. The remaining UIEs available to the user are the same as in the previous view of the Demo App (Fig. 5.20).

- **Perspective Correction Button (bottom view, right)** – The main control UIE of the detected corners view is the Perspective Correction Button, which enables the user to initiate the application of a *perspective transform* on the filtered image of the frame, as well as the pattern's. This image transformation heavily

relies on both the accurate detection of the corners, and their correct assignment to the four quadrants of the frame they *should* belong to, and is the main topic discussed in this subsection.

The algorithm performing the perspective transform on the filtered images of the frame and the pattern, was developed in the Swift programming language for iOS, based on Austin (n.d.) and Blinn (2003). The entries of the transformation matrix applied on the images, depend on two sets of *four* points each, called *source* and *destination points*. Both sets must be specified in pixels, and every point in one set, must have an established one-to-one correspondence with a point in the other set. In principle, the source points can be any set of corresponding corners of the frame earlier *detected*, that is, the four arc beginnings, the four arc outer endings, or the four arc inner endings. Since the corner detection algorithm consistently detects *all* of the frame's corners, this implementation will make use of the arc beginnings. Note that the positions of the frame's corners given in polar coordinates by Eqs. 3.43, still refer to the plot coordinate system, and cannot therefore establish a correspondence with the detected corners specified in pixels.

The reason behind the choice of the arc beginnings as the source points, is simply convenience; in the original, distortion-free frame of the NLCode, those corners form an upright square. Based on this observation, defining a square around the center of any of the processed images, will provide another set of four corners which, if reasonably chosen, can be thought of as the *correct* locations of the frame's corners, i.e. if no distortions were present. As soon as the above square is assigned with a reasonable size, its corners will be used as the *destination* points required for the computation of the perspective transform matrix.

The size of the square is defined via its half-diagonal, which is set equal to the mean of the radii computed in Subsec. 5.2.1. It is reminded that starting from the inner and outer radii of the target area, the frame was iteratively "*closed in on*" from the inside and the outside, in order to define the masks separating the two main elements of the NLCode. The values of the maximum inner $R_{inner,max}^{TA, Pixels}$ and minimum outer $R_{outer,max}^{TA, Pixels}$ radius of the target area were given in Fig. 5.3, and their mean is 653 *Pixels*.

Figure 5.23 shows the results of the perspective correction on the NLCode images, including the final image of the pattern, which will be used for its tracing in order to retrieve of the 2D projected trajectory.

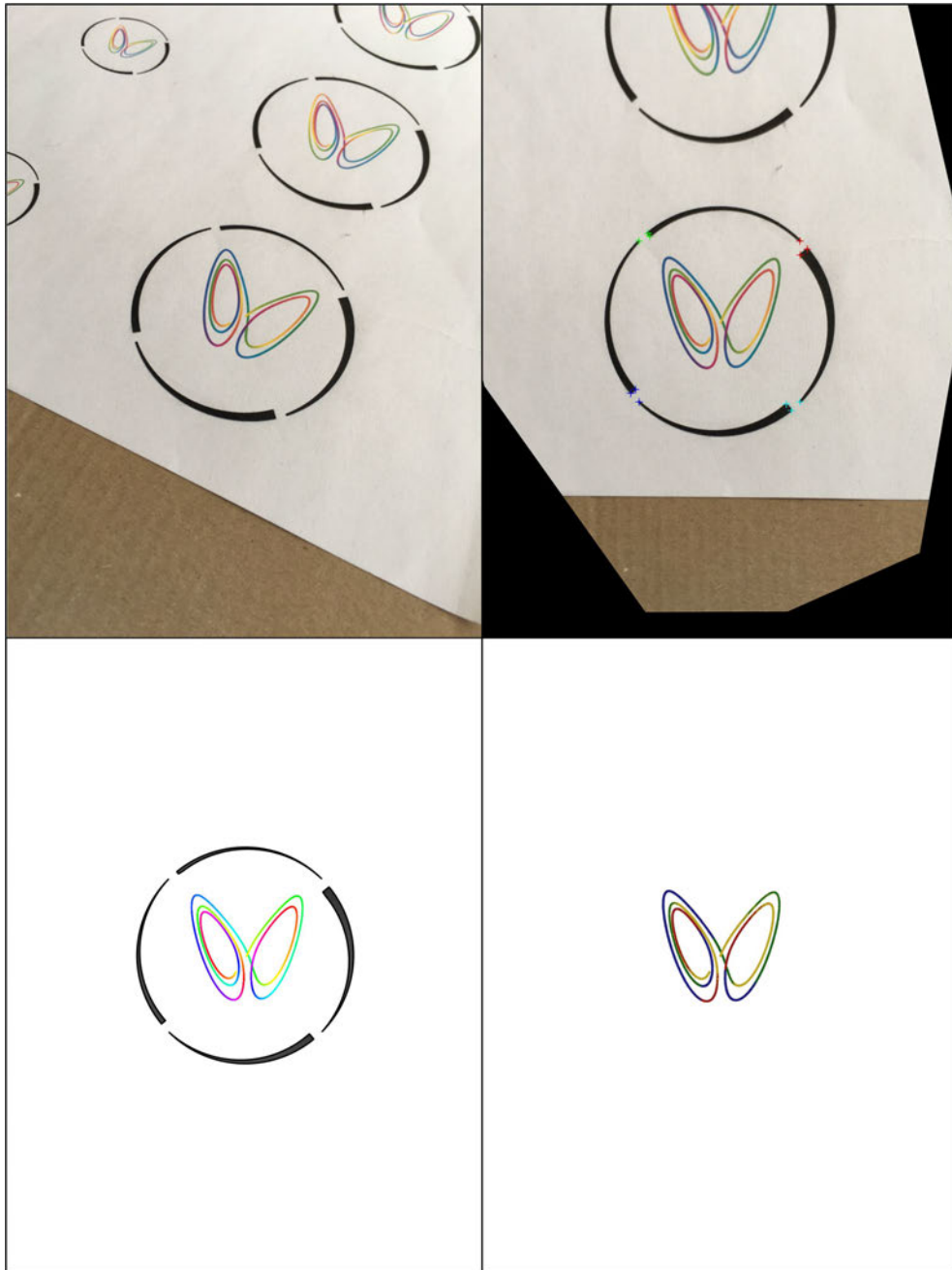


Figure 5.23: Perspective correction on the NLCode's images – Top Left:: the captured photograph of the NLCode, prior to any processing. Top Right: The same image, after the application of the perspective transform, with the detected corners of the frame, transformed by the same matrix. Bottom Left: The digital version of the NLCode, before printing. Bottom Right: The masked and filtered image of the pattern, after perspective correction. This is the image the tracing algorithm will be applied on.

5.3 NLCode Reading

Figure 5.24 shows the sixth view of the Demo App. This view is called *perspective corrected view* because it displays the image of the NLCode's nonlinear pattern, after the perspective transform presented in Subsec. 5.2.4 was applied to it (Fig. 5.23, bottom-right). The UI on the detected corners view presents the user with a few options, each corresponding to a UIE displayed on the view.

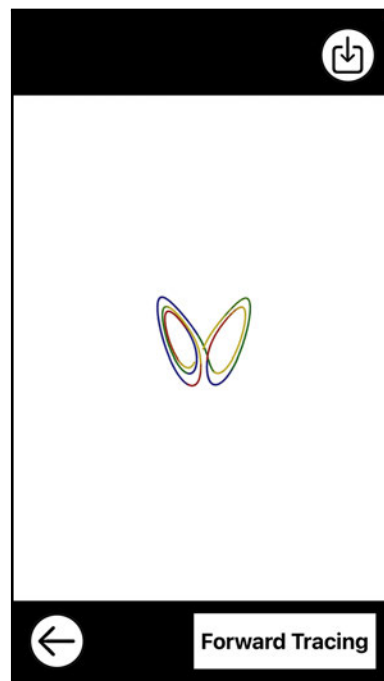


Figure 5.24: Demo App - Perspective Corrected View & Forward Tracing prompt: The sixth view of the Demo App shows the image of the NLCode's pattern, after the perspective transform was applied to it (Fig. 5.23, bottom-right). The Forward Tracing Button displayed at the bottom-right of the view, prompts the user to proceed with the processing of the displayed image. When the Forward Tracing Button is tapped, the Demo App applies on the image a forward tracing algorithm, which traces the NLCode's pattern until its end. The remaining UIEs available to the user are the same as in the previous view of the Demo App (Fig. 5.22).

- **Forward Tracing Button (bottom view, right)** – The main control UIE of the perspective corrected view is the Forward Tracing Button, which enables the user to initiate the application of a *forward tracing algorithm* on the corrected image of the pattern. This algorithm integrates a trajectory forward in time, as it evolves toward the 2D projected trajectory of the NLCode, at which point it couples with it and starts tracing it.

The NLCode captured depicts a 2D projected trajectory on the x^1x^3 plane. According to [Subsec. 3.3.7](#), the coupling component driving the response subsystem is x^1 . This component must be retrieved from the processed image of the nonlinear pattern of the NLCode. The forward tracing algorithm begins solving a 3D transformed Lorenz system, with random ICs taken from the basin of attraction, in plot coordinates. [Subsec. 3.3.6](#). At every step of the RK4 method, the tracing converts the current 2D projected state to pixel coordinates, and checks the colour of the current pixel. For as long as the pixels checked are white, the solution continues without a drive. Note that the solution will eventually cross paths with the pattern, since the evolving trajectory is drawn to the attractor. The first pixel of the pattern found on the path of the trajectory is used as a reference, in search of adjacent pixels of the pattern toward the general direction of the solution. If such a pixel is found, the solution ignores its own \tilde{x}^1 component in favour of the x^1 component that corresponds to the pixel of the pattern, that is, it is being driven as the response subsystem of the composite [Sys. 3.60](#). As can be seen on the left image of [Fig. 5.25](#), the solution is eventually captured not simply by the attractor, but by the nonlinear pattern on the image. When this happens, the response subsystem starts synchronising with the drive, and starts tracing it. This process continues until the end of the pattern. The response takes a few steps beyond the end of the pattern in order to confirm its ending and when it does, traces those extra steps back, and hands over the plot coordinate at the end of the pattern to the next process.

[Figure 5.25](#) (left) shows the seventh view of the Demo App. This view is called *forward tracing view* because it displays the image of the NLCode's nonlinear pattern, with the steps taken by the response subsystem during the forward tracing. For illustration purposes, the trace drawn in green crosses only shows every other 10 iteration steps of the evolving trajectory.

- **Backward Reading Button (bottom view, right)** – The main control UIE of the forward tracing view is the Backward Reading Button, which enables the user to initiate the application of a *backward tracing algorithm* on the image of the pattern. This algorithm integrates a trajectory backward in time, and since it starts on the nonlinear pattern, it is continuously driven and maintains the synchronisation with the pattern.

The backward tracing algorithm performs the actual reading of the NLCode. The system solved this time, is the proper 2D response subsystem of [Sys. 3.60](#). The ICs are taken from the forward tracing algorithm, and since they are already on the pattern, the algorithm traces it. This means that the backward tracing has a constant supply of driving values, which keep it synchronised with the pattern. As soon as the backward evolving trajectory reaches the beginning of the pattern it stops, and hands over the traced trajectory to the next process.

[Figure 5.25](#) (right) shows the eighth view of the Demo App. This view is called *request authentication view* because it displays the image of the NLCode's nonlinear pattern, with the steps taken by the response subsystem during the backward reading. The trace drawn in red crosses only shows every other 10 iteration steps of the evolving trajectory, for illustration purposes.

- **Request Authentication Button (bottom view, right)** – The main control UIE of the backward tracing view is the Request Authentication Button which, were it implemented, could enable the user to initiate an authentication process, by sending the sequence of numbers (plot coordinates) that corresponds to the traced pattern to a database server, according to what was discussed in [Subsec. 1.3.3](#), and illustrated in [Fig. 1.3](#).

Two important remarks regarding the tracing and reading processes described and visually demonstrated in this subsection, are the following: Both forward and backward algorithms go back and forth between plot and pixel coordinates, in every iterative step of the RK4 method. The transformation between those two coordinate systems was calculated using the method of least squares adjustment presented in [Subsec. 3.2.2](#), using as observation points the detected and identified corners of the frame (pixel coordinates), and as control points the known plot coordinates of the same corners on the plotted version of the NLCode.

The second important note is that not all plot points in the backward traced trajectory are usable. As can be seen in both images of [Figure 5.25](#), even skipping 10 points between the crosses drawn, there are still several points that are too close to one another to be considered distinct. The reason for that is twofold. Firstly, due to the dynamics of the underlying system, the distances traversed in equal time intervals, are not equal themselves. This is evident by the presence of densely and sparsely populated segments of the pattern. Secondly, the iteration step of the RK4 method is too small ($\Delta t = 10^{-3}$ sec) to create a sequence of points, all of which can be meaningfully represented on the discrete space of a

digital image. This means that the sequence of numbers returned at the end of the backward tracing, the *reading* of the NLCode, should be processed by an agreed upon method between sender and receiver, in order to facilitate an authentication process.

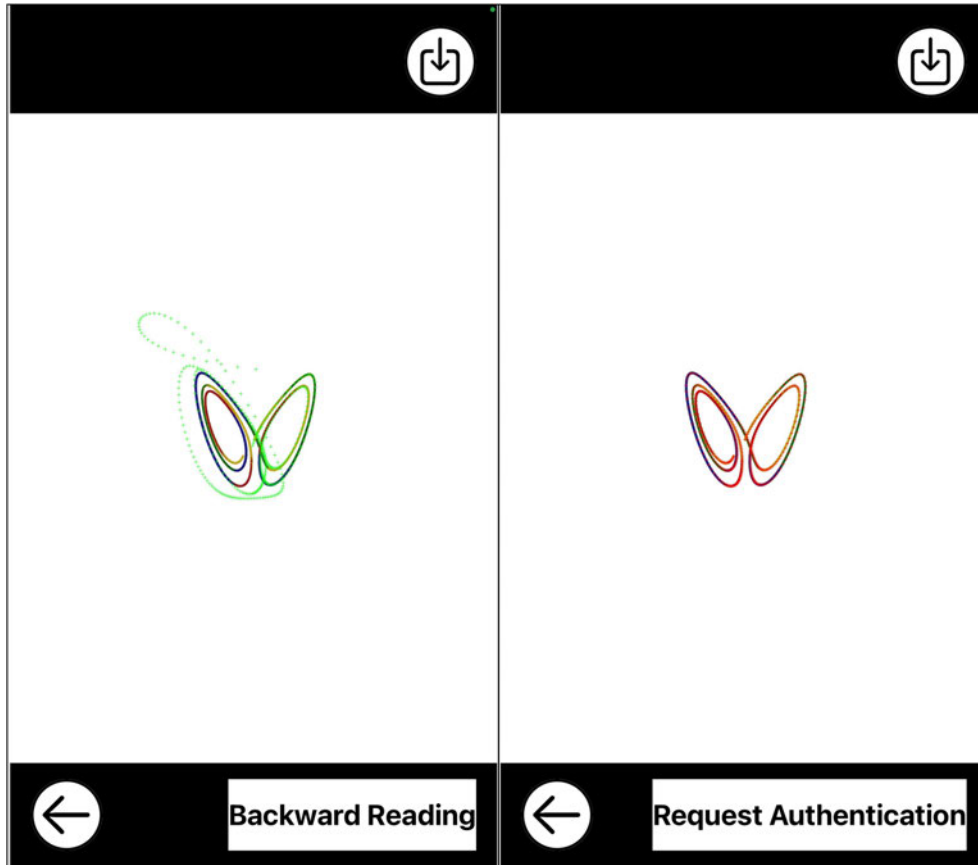


Figure 5.25: Demo App - Forward & Backward Tracing Views: The image on the left is the image of the pattern, with the points traced by the forward tracing algorithm drawn as green crosses. The solution of the response system begins from a point taken at random from the basin of attraction, and wanders around the image until it is captured by the pattern. Driven by the pattern's x^1 component, it starts synchronising with it. By the time it reaches the end of the pattern, the evolving trajectory coincides with the pattern, and is ready to hand over its last state to the backward reading algorithm. The image on the right shows the new trajectory evolving backward in time, while staying synchronised with the driving component of the pattern, until it reaches its end, having obtained a sequence of traced points on the image. That sequence can be further processed, or sent directly to a database server for authentication, by tapping the Request Authentication Button.

6

Summary & Conclusion

This thesis began with a statement of the problem the underlying project was called to address, that is, counterfeiting. The introduction presented the main objectives of the thesis, and gave an overview of the counterfeiting problem from a socioeconomic perspective. The official Literature Review for this project included the gathering of anti-counterfeiting technologies suited to the sponsor company's manufacturing process, and their evaluation in terms of this, as well as other criteria. A quick look at recent trends revealed a preference toward anti-counterfeiting technologies that utilise the fast growing capabilities of smartphone devices, in order to offer user-based product authentication. This first part of the thesis then turned to the proposed technology. The NLCode scheme was given a brief overview, focusing on its main components and operation principle, i.e. that of chaotic synchronisation.

The second chapter gave a brief introduction to those concepts of Nonlinear Dynamics pertinent to the proposed scheme. This was the *"unofficial"* Literature Review of this project. This chapter discussed the numerical solution of chaotic systems, a fundamental property possessed by some of them, called dissipation, and their equilibrium solutions. It also introduced the Lyapunov exponents, which form the basic criterion for the synchronisation of chaotic systems. Attractors and their basins, are both central in the development of the scheme. These were briefly reviewed, before a presentation of the type of chaotic synchronisation the developed scheme relies on, i.e. that occurring between subsystems forming homogeneously driving composite systems, with very clearly defined properties.

Part I of this thesis was dedicated to the structural elements of the NLCode. The arced circular frame was thoroughly discussed, and its parameters were defined, listed and specified. The variable transformations between system and plot, and plot and pixel coordinates were discussed in detail. The former gives a unified approach to the scheme, allowing the implementation of different dynamical systems, under the coordinate system defined by the frame of the feature. The latter bridges events related to the Dynamics of the trajectories, to events taking place on the captured image of the NLCode.

The thesis then turned to the Lorenz system, which formed the basis for the development of the scheme. All the concepts and methods introduced in the theory chapter, were applied on the Lorenz system, in preparation for their later use in the development of the Generator and the Reader of the feature. Lastly, the colour profile of the NLCode was introduced and briefly examined.

The NLCode Generator is maybe the most challenging part of this project, which makes sense, since it is responsible for producing readable NLCodes. This means that it has to anticipate numerous issues that might be faced along the way and treat them accordingly. The Generator treats the candidate trajectories for spatial and colour overlaps, both of which are crucial to the performance of the scheme. The operating window of the NLCode, in terms of print sizes and working distances was also theoretically estimated and provided valuable insight in certain shortcomings of the test photographic sample later used for the development of colour filters.

The NLCode Reader was presented using the Demo App developed for the retrieval of the trajectory depicted on the feature. From the target area facilitating and enabling several processes following it, to the also challenging and quite extensive section on image processing. The latter dealt with image masking, colour detection and filtering, it employed a corner detector and implemented a perspective correction shader.

The thesis concluded with the forward tracing and backward reading of the nonlinear pattern of the NLCode. Making use of the synchronisation property of chaotic systems, these processes were capable to retrieve the trajectory printed on the feature, and did so in a visually pleasing manner.

Appendix: The Dequan Li System

The Dequan Li system does not model any particular phenomenon. This system was synthesised in 2006 by Dequan Li in a successful attempt to fill a gap in literature, by showing that a 3D autonomous chaotic dynamical system, “with smooth quadratic terms”, can possess a *three-scroll* attractor (D. Li, 2008, p. 388). Up until that point, this type of systems were known to possess one-scroll attractors, such as the Rössler system (Rössler, 1976), two-scroll attractors like the Lorenz attractor seen in Fig. 3.4 (E. Lorenz, 1963), and the four-scroll attractor of the Lü-Chen system (Lü et al., 2004). It is interesting to note that, after the discovery of the Lorenz attractor, the motivation behind the pursuit of systems with a wide range of properties was not only theoretical, but practical as well, aiming to advance “various chaos-based technologies and information systems” (D. Li, 2008, p. 388).

The Dequan Li system is mathematically described by the following 3D system of 1st order ODEs, which is algebraically reminiscent of the Lorenz system (Sys. 3.45), with two additional quadratic terms in the first and third equations,

$$\begin{aligned}\dot{\chi}^1 &= a(\chi^2 - \chi^1) + d\chi^1\chi^3 \\ \dot{\chi}^2 &= k\chi^1 + f\chi^2 - \chi^1\chi^3 \\ \dot{\chi}^3 &= -e(\chi^1)^2 + \chi^1\chi^2 + c\chi^3.\end{aligned}\tag{A.1}$$

The system parameters in this case are six, namely a , d , k , f , e , and c , and they are all positive.

A.1 Numerical Solution

Figure A.1 shows a 3D trajectory of the Dequan Li system (Sys. A.1), solved using the RK4 method presented in Sec. 2.1 with the parameter values used in Letellier and Gilmore (2008)³² and ICs taken from (D. Li, 2008), i.e.

$$a = 41, \quad d = 0.16, \quad k = 55, \quad f = 20, \quad e = 0.65, \quad c = 11/6 \tag{A.2a}$$

$$\chi_0^1 = 2.0, \quad \chi_0^2 = 2.0, \quad \chi_0^3 = 2.0. \tag{A.2b}$$

³²Letellier and Gilmore (2008) pointed out that for the first parameter value $a = 40.0$, as Dequan Li had suggested (D. Li, 2008), the attractor is actually a limit cycle. This was partially confirmed by the Lyapunov exponents calculated in this study for this value of a , which gave a maximal Lyapunov exponent close to zero, and two more negative exponents. Setting $a = 41.0$ after Letellier and Gilmore (2008) gives a positive, a zero and a negative exponent (Sec. A.5, Eq. A.12) which, according to Table 2.2, is consistent with the presence of a chaotic attractor.

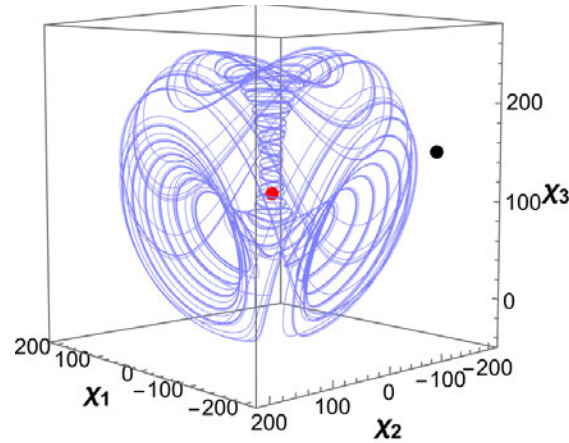


Figure A.1: Dequan Li attractor in 3D before the transformation: Solution of the Dequan Li system (Sys. A.1) using the RK4 scheme presented in section Sec. 2.1. The parameter values and ICs are the ones used in Letellier and Gilmore (2008) and D. Li (2008) respectively (Eqs. A.2). The iteration step is $\Delta t = 10^{-3}$ seconds and the total number of iterations $N = 20,000$.

A.2 Variable Transformation

The transformed form of the Dequan Li system is obtained similarly to the two previous cases, after applying on Sys. A.1 the one-step transformation given by Eq. 3.24, and then simplifying by letting $S^1 = S^2 = S^3 = 1/S$ (Eq. 3.19):

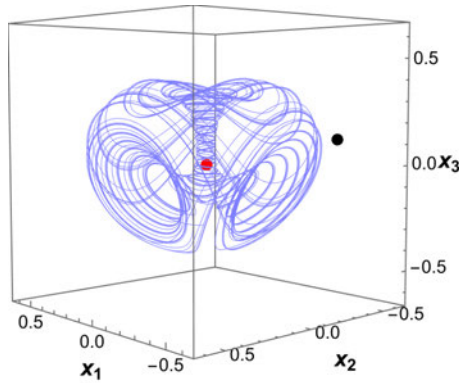
$$\begin{aligned}\dot{\mathbb{X}}^1 &= \frac{1}{T} [a (\mathbb{X}^2 - \mathbb{X}^1) + d S \mathbb{X}^1 \mathbb{X}^3] \\ \dot{\mathbb{X}}^2 &= \frac{1}{T} (k \mathbb{X}^1 + f \mathbb{X}^2 - S \mathbb{X}^1 \mathbb{X}^3) \\ \dot{\mathbb{X}}^3 &= \frac{1}{T} [-e S (\mathbb{X}^1)^2 + S \mathbb{X}^1 \mathbb{X}^2 + c \mathbb{X}^3],\end{aligned}\tag{A.3}$$

where once more, $(\mathbb{X}^1, \mathbb{X}^2, \mathbb{X}^3) = \mathbb{X} = \mathbf{x} - \mathbf{c} + (\boldsymbol{\chi}_c/S)$ is the “collective variable” introduced in Eq. 3.48 for the simplification of the algebraic form of the system.

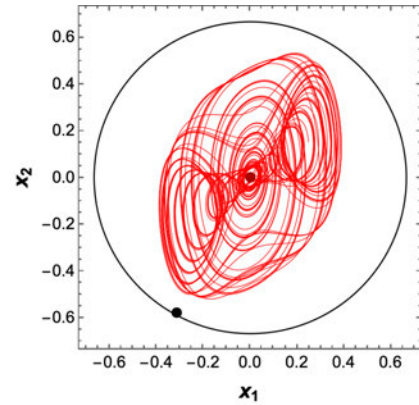
The calculation of the center $\boldsymbol{\chi}_c$ of the Dequan Li attractor, and of the farthest point $\boldsymbol{\chi}_f$ of the attractor from its center, was performed as for the Lorenz system (Sec. 3.3, Subsec. 3.3.2), with a number of $N = 100,000$ points in the trajectory. The results are given in the following set of equations and also shown in Fig. A.1:

$$\boldsymbol{\chi}_c = (-1.2837, -0.9355, 107.5150)\tag{A.4a}$$

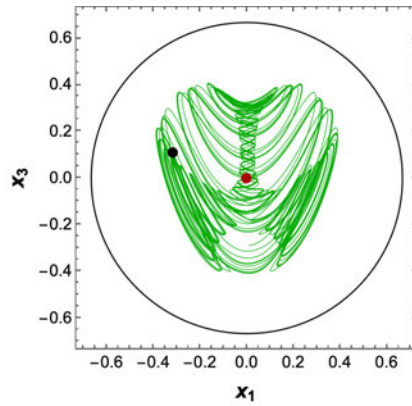
$$\boldsymbol{\chi}_f = (-116.756, -211.495, 148.410).\tag{A.4b}$$



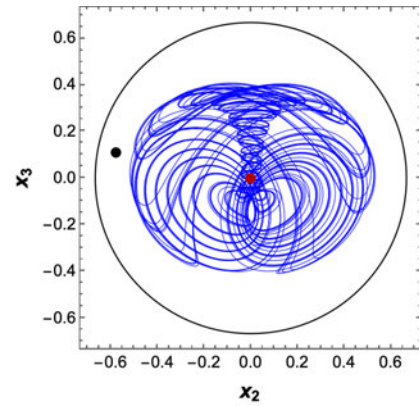
(a) Scaled and translated Dequan Li attractor in 3D.



(b) Projection of the 3D trajectory on the x^1x^2 plane.



(c) Projection of the 3D trajectory on the x^1x^3 plane.



(d) Projection of the 3D trajectory on the x^2x^3 plane.

Figure A.2: Dequan Li attractor in 3D and 2D projections, after the transformation: (a) Solution of the transformed Dequan Li system (Sys. A.3) using the RK4 scheme presented in section Sec. 2.1. The parameter values are the same as before (Fig. A.1), and the ICs are the transformed versions of the original ones given in Eq. A.2b, specifically, $x_0 = (0.008986, 0.008034, -0.288764)$. The iteration step and the total number of iterations are the same as in Fig. A.1. The plot also shows in red the transformed center $x_c = (0.0, 0.0, 0.0) = c$ of the attractor (Eq. 3.24), and in black the transformed farthest point $x_f = (-0.3160, -0.5762, 0.1119)$ from the center. (b), (c), and (d) are the 2D projections of the trajectory, and points x_c and x_f shown in Fig. A.2a, on the x^1x^2 , x^1x^3 , and x^2x^3 planes respectively. These plots also include the cross-sections (meridians) of the enclosing sphere of radius $R_{inner} = 2/3$ (Table 3.1) cut by the three planes. These plots further demonstrate the point that regardless of whether the two-dimensional projected x_f is farthest from the projected x_c or not, the cross-sectioned sphere by the respective plane will always encircle the attractor entirely (also see Fig. 3.5).

The scaling parameter S was calculated using [Eq. 3.30](#) which led to

$$S = 365.402. \quad (\text{A.5})$$

[Figure A.2](#) shows the Dequan Li attractor after the scaling and translation transformations, and its three 2D projections on the x^1x^2 , x^1x^3 , and x^2x^3 planes. The scaled and translated versions x_c and x_f of the center and farthest point from center respectively are also shown in both 3D and 2D.

A.3 Dissipation

The case of the Dequan Li system is not as straightforward as the Lorenz system in terms of confirming dissipation. The stability matrix ([Eq. 2.8](#)) of the transformed Dequan Li system, [Sys. A.3](#), is

$$\mathcal{A}(\mathbb{X}) = \frac{1}{T} \cdot \begin{pmatrix} d S \mathbb{X}^3 - a & a & d S \mathbb{X}^1 \\ k - S \mathbb{X}^3 & f & -S \mathbb{X}^1 \\ S \mathbb{X}^2 - 2e S \mathbb{X}^1 & S \mathbb{X}^1 & c \end{pmatrix}, \quad (\text{A.6})$$

and its trace includes one of \mathbb{X} 's components, \mathbb{X}^3 , which makes the divergence of the system's vector field ([Eq. 2.14](#)),

$$\nabla \cdot \mathbf{f} = \frac{1}{T} (d S \mathbb{X}^3 - a + c + f). \quad (\text{A.7})$$

This means that, for the Dequan Li system to be dissipative, i.e. $\nabla \cdot \mathbf{f} < 0$, the \mathbb{X}^3 component should be less than $(a - c - f)/(d S)$ or, using [Eq. 3.48](#),

$$x^3 < \frac{a - c - f + d S c^3 - d \chi_c^3}{d S}. \quad (\text{A.8})$$

Using the parameter values given by [Eqs. A.2a, A.4a](#) and [A.5](#), as well as [Table 3.1](#) where it is implied that the attractor should be centred at $\mathbf{c} = \mathbf{0}$, it is seen that $\nabla \cdot \mathbf{f} < 0$ only in the region of state space where $x^3 < 0.0336$, and non-negative elsewhere. Moreover, comparing this value against any of the plots in [Fig. A.2](#) showing an x^3 -axis, it becomes clear that roughly half of the Dequan Li attractor falls in a region of state space with non-negative divergence. This, in fact, is the issue mentioned in [Sec. 2.2 \(footnote 12\)](#) regarding the sole use of the *divergence*

criterion to establish dissipation in a dynamical system. Dequan Li encountered it while examining the basic properties of his synthesised system (Sys. A.1), and used the Lyapunov dimension D_L (Kaplan & Yorke, 1979) to argue that, since $2 < D_L = 2.1165 < 3$, the system must be dissipative – were it conservative, its trajectories would fill a 3D volume (D. Li, 2008).

A.4 Equilibrium Solutions and Local Stability

For the set of system parameters used here (Eq. A.2a), the Dequan Li system has three equilibrium solutions in total, but *only one of them is real*. Just as all of the system's equilibrium solutions, the fixed point \mathbf{x}_\circ^* is the solution of Eq. 2.15, for $\mathbf{f}(\mathbf{x}^*)$ equal the vector field defined by Sys. A.3:

$$\mathbf{x}_\circ^* = \mathbf{c} - \frac{\mathbf{X}_c}{S}. \quad (\text{A.9})$$

For the original, untransformed Dequan Li system (Sys. A.1), the fixed point given above is the origin and the “collective variable” $\mathbf{X}_\circ^* = \mathbf{0}$. The stability matrix (Eq. A.6) at \mathbf{X}_\circ^* is

$$\mathcal{A}(\mathbf{X}_\circ^*) = \frac{1}{T} \cdot \begin{pmatrix} -a & a & 0 \\ k & f & 0 \\ 0 & 0 & c \end{pmatrix}. \quad (\text{A.10})$$

The eigenvalues of $\mathcal{A}(\mathbf{X}_\circ^*)$ are found by solving Eq. 2.19:

$$\lambda_\circ^* = \frac{c}{T} \quad (\text{A.11a})$$

$$\lambda_{\circ\pm}^* = \frac{-(a-f) \pm \sqrt{(a+f)^2 + 4ak}}{2T}. \quad (\text{A.11b})$$

Similarly to the Lorenz system, all three of the above eigenvalues are *real*. The first and the second eigenvalues are positive, that is, $\lambda_\circ^* > 0$ and $\lambda_{\circ+}^* > 0$, and the third eigenvalue is negative, i.e. $\lambda_{\circ-}^* < 0$. With “at least one eigenvalue with a positive real part, and one with a negative real part”, according to Sec. 2.2 (also see Table 2.1), \mathbf{x}_\circ^* is an *unstable* equilibrium solution of the *saddle* type.

The above results are provided in numerical form, i.e. after substituting the parameter values of Eq. A.2a into all relevant equations, in Table A.1.

Table A.1: Stability of the real equilibrium solution of the Dequan Li system. The first column lists the one fixed point of *Sys. A.3* that is real (Eq. A.9), after substituting the parameter values from Eq. A.2a. The second column gives the eigenvalues of the stability matrix calculated at x_{\circ}^* , found using Eqs. A.11. The third column shows the stability type of the fixed point, based on the signs of the real parts of the eigenvalues, and according to the criteria presented in Sec. 2.2 and summarised in Table 2.1.

| Equilibrium Point x^* | Eigenvalues of $\mathcal{A}(x^*)$ | Stability Type |
|---|---|----------------------|
| $x_{\circ}^* = (3.5130 \times 10^{-3},$ $2.5603 \times 10^{-3},$ $-0.2942)$ | $\lambda_{\circ}^* = 1.8333$ $\lambda_{\circ+}^* = -66.9380$ $\lambda_{\circ-}^* = 45.9380$ | Saddle (unstable) |

A.5 Lyapunov Spectrum & Lyapunov Dimension

The Lyapunov spectrum of the Dequan Li attractor was calculated using the same software library *DynamicalSystems.jl* (see Sec. 2.4), and was found to be

$$L_1 = 0.487, \quad L_2 = 0.0, \quad L_3 = -2.371. \quad (\text{A.12})$$

Based on the above results, the Lyapunov dimension is calculated from Eq. 2.33 and turns out to be

$$D_L = 2.206. \quad (\text{A.13})$$

With D_L a non-integer and one positive, one zero and one negative Lyapunov exponent (see Table 2.2, Sec. 2.6), the Dequan Li attractor is confirmed to be both strange and chaotic.

A.6 Basin of Attraction

Figure A.3 shows the basin of attraction (Sec. 2.5) of the Dequan Li attractor. This set appears to extend to infinity, like the basin of the Lorenz attractor, but also seems featureless. This is an example where the lack of a pool of ICs *might not* prove to be detrimental to the search of suitable trajectories by the NLCode Generator, since all, or *almost all* ICs taken from the large vicinity of the attractor would quite possibly result in trajectories captured by it.

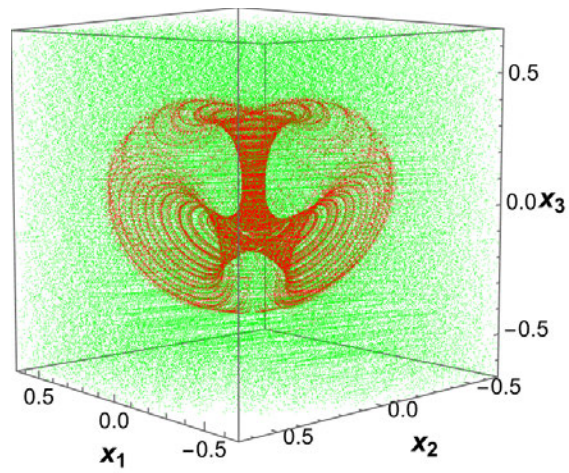


Figure A.3: Basin of attraction of the Dequan Li attractor. The Dequan Li attractor is shown in red, and its basin of attraction in green. Like the Lorenz attractor, the basin of attraction of the Dequan Li attractor seems to also extend to infinity, but unlike the basin Lorenz attractor it is quite featureless.

Bibliography

- Andor Oxford Instruments. (2017). What is spatial resolution & pixel size in CCD cameras? Date accessed: 02-04-2024. <https://andor.oxinst.com/learning/view/article/ccd-spatial-resolution>
- Apple Inc. (2012). Documentation Archive: Drawing and Printing Guide for iOS – iOS Drawing Concepts. Date Accessed: 02-04-2024. <https://developer.apple.com/library/archive/documentation/2DDrawing/Conceptual/DrawingPrintingiOS/GraphicsDrawingOverview/GraphicsDrawingOverview.html>
- Apple Inc. (2021). Creating App Clip Codes | Apple Developer Documentation. Date Accessed: 02-04-2024. https://developer.apple.com/documentation/app_clips/creating_app_clip_codes#
- Austin, D. (n.d.). Using Projective Geometry to Correct a Camera, Feature Column, AMS. Date Accessed: 02-04-2024. <https://www.ams.org/publicoutreach/feature-column/fc-2013-03>
- Baldini, G., & Pons, E. C. (2017). Enforcers and brand owners' empowerment in the fight against counterfeiting. *JRC Publications Repository*, 55.
- Benettin, G., Galgani, L., Giorgilli, A., & Strelcyn, J. M. (1980a). Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. Part 1: Theory. *Meccanica*, 15, 9–20.
- Benettin, G., Galgani, L., Giorgilli, A., & Strelcyn, J. M. (1980b). Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; A method for computing all of them. Part 2: Numerical application. *Meccanica*, 15, 21–30.
- Blinn, J. (2003). *Jim Blinn's corner : notation, notation, notation*. Morgan Kaufman Publishers.
- Chesterton, G. (2007). Tremendous Trifles, XVII. The Red Angel. Dover Publications.
- Couch II, L. W. (2013). *Digital and Analog Communication Systems* (8th ed.). Pearson.
- Credence Research. (2018). Anti-counterfeit Packaging Technologies Market by Type, by End-use Industry – Growth, Share, Opportunities & Competitive Analysis, 2018–2026. Date Accessed: 23-05-2022. <https://www.credenceresearch.com/report/anti-counterfeit-packaging-technologies-market>

- Cuomo, K. M. (1994). Analysis and Synthesis of Self-Synchronizing Chaotic Systems. *PhD Thesis*, 228.
- Cuomo, K. M., & Oppenheim, A. V. (1993). Chaotic signals and systems for communications. *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 3.
- Cvitanovic, P., Artuso, R., Mainieri, R., Tanner, G., & Vattay, G. (2020). *Chaos: Classical and Quantum* (17th ed.). Niels Bohr Institute, Copenhagen.
- Datseris, G. (2018). DynamicalSystems.jl: A Julia software library for chaos and nonlinear dynamics. *Journal of Open Source Software*, 3, 598.
- De Leeuw, K. M. M., & Bergstra, J. (2007). *The History of Information Security: A Comprehensive Handbook*. Elsevier.
- Devaney, R. (1989). *An Introduction To Chaotic Dynamical Systems* (2nd ed.). Addison - Wesley.
- Device Specifications. (n.d.). Apple iPhone 6s Plus - Specifications. Date Accessed: 02-04-2024. <https://www.devicespecifications.com/en/model/13ea3678>
- Ditto, W. L., Spano, M. L., Savage, H. T., Rauseo, S. N., Heagy, J., & Ott, E. (1990). Experimental observation of a strange nonchaotic attractor. *Physical Review Letters*, 65, 533.
- Eckmann, J. P., & Ruelle, D. (1985). Ergodic theory of chaos and strange attractors. *Reviews of Modern Physics*, 57, 617–656.
- Fairchild, M. D. (2013). *Color Appearance Models*. John Wiley & Sons, Ltd.
- Farmer, J. D., Ott, E., & Yorke, J. A. (1983). The dimension of chaotic attractors. *Physica D: Nonlinear Phenomena*, 7, 153–180.
- Feudel, U., Kuznetsov, S., & Pikovsky, A. (2006, April). *Strange Nonchaotic Attractors - Dynamics between Order and Chaos in Quasiperiodically Forced Systems* (Vol. 56). WORLD SCIENTIFIC.
- Frederickson, P., Kaplan, J. L., Yorke, E. D., & Yorke, J. A. (1983). The liapunov dimension of strange attractors. *Journal of Differential Equations*, 49, 185–207.
- Gaiman, N., & McKean, D. (2012). *Coraline* (N. Gaiman- Author & D. McKean-Illustrator, Eds.; 10th Anniversary). HarperCollins.
- Gan, G., Ma, C., & Wu, J. (2007). *Data clustering : Theory, Algorithms, and Applications*. Society for Industrial & Applied Mathematics (SIAM).

- Gan, X., Wang, H., Yuan, R., & Ao, P. (2021). A New Criterion Beyond Divergence for Determining the Dissipation of a System: Dissipative Power. *Frontiers in Physics, 9*, 458.
- Geist, K., Parlitz, U., & Born, W. L. (1990). Comparison of Different Methods for Computing Lyapunov Exponents. *Progress of Theoretical Physics, 83*, 875–893.
- Ghilani, C. D. (2017). *Adjustment computations : spatial data analysis* (6th ed.). John Wiley & Sons.
- Goldstein Research. (2019). Global Metal Packaging Market Analysis by product, by material & by end-user with regional outlook & Forecast to 2025. Date Accessed: 23-05-2022. <https://www.goldsteinresearch.com/report/global-metal-packaging-market-analysis>
- Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing* (3rd ed.). Pearson.
- Grebogi, C., Ott, E., Pelikan, S., & Yorke, J. A. (1984). Strange attractors that are not chaotic. *Physica D: Nonlinear Phenomena, 13*, 261–268.
- Grobman, D. M. (1959). Homeomorphisms of systems of differential equations. *Doklady Acad. Nauk SSR, 128*, 880–881.
- Guckenheimer, J., & Holmes, P. (1983). *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields* (Vol. 42). Springer New York.
- Haken, H. (1983). At least one Lyapunov exponent vanishes if the trajectory of an attractor does not contain a fixed point. *Physics Letters A, 94*, 71–72.
- Hamming, R. W. (W. (1973). *Numerical methods for scientists and engineers* (2nd ed.). Dover Publications.
- Hartman, P. (1963). On the local linearization of differential equations. *Proceedings of the American Mathematical Society, 14*, 568–573.
- Hearn, D. D., Baker, M. P., & Carithers, W. R. (2010). *Computer Graphics with Open GL* (4th ed.). Pearson.
- Hirsch, M. W., Smale, S., Devaney, R. L., & Hirsch, M. W. (2004). *Differential equations, dynamical systems, and an introduction to chaos*. Elsevier/Academic Press.
- IEC. (1999). IEC 61966-2-1:1999 – Multimedia systems and equipment – Colour measurement and management – Part 2-1: Colour management – Default RGB colour space – sRGB | IEC Webstore. Date Accessed: 02-04-2024. <https://webstore.iec.ch/publication/6169>

- ISO. (2006). ISO/IEC 16022:2006 - Information technology — Automatic identification and data capture techniques — Data Matrix bar code symbology specification. Date Accessed: 07-01-2024. <https://www.iso.org/standard/44230.html>
- ISO. (2008). ISO/IEC 24778:2008 - Information technology — Automatic identification and data capture techniques — Aztec Code bar code symbology specification. Date Accessed: 07-01-2024. <https://www.iso.org/standard/41548.html>
- ISO. (2012). ISO 12931:2012 (Withdrawn, revised by ISO 22383:2020) - Performance criteria for authentication solutions used to combat counterfeiting of material goods. Date Accessed: 31-07-2020. <https://www.iso.org/standard/52210.html>
- ISO. (2015). ISO/IEC 18004:2015 - Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification. Date Accessed: 07-01-2024. <https://www.iso.org/standard/62021.html>
- ISO. (2020). ISO 22383:2020 - Security and resilience — Authenticity, integrity and trust for products and documents — Guidelines for the selection and performance evaluation of authentication solutions for material goods. Date Accessed: 02-04-2024. <https://www.iso.org/standard/50285.html>
- Jähne, B. (2004). *Practical Handbook on Image Processing for Scientific and Technical Applications* (2nd ed.). CRC Press.
- Kaplan, J. L., & Yorke, J. A. (1979). Chaotic behavior of multidimensional difference equations. In H.-O. Peitgen & H.-O. Walther (Eds.). Springer-Verlag Berlin Heidelberg NewYork.
- Katok, A., & Hasselblatt, B. (1995, April). *Introduction to the Modern Theory of Dynamical Systems*. Cambridge University Press.
- Larson, B. (2019, June). GPUImage2: A BSD-licensed Swift framework for GPU-accelerated video and image processing. Sunset Lake Software. Date Accessed: 12-02-2024. <https://github.com/BradLarson/GPUImage2>
- Leal, L. G. (2007). *Advanced Transport Phenomena: Fluid Mechanics and Convective Transport Processes*. Cambridge University Press.
- Letellier, C., & Gilmore, R. (2008). Poincaré sections for a new three-dimensional toroidal attractor. *Journal of Physics A: Mathematical and Theoretical*, 42, 015101.

- Li, D. (2008). A three-scroll chaotic attractor. *Physics Letters, Section A: General, Atomic and Solid State Physics*, 372, 387–393.
- Li, L. (2013). Technology designed to combat fakes in the global supply chain. *Business Horizons*, 56, 167–177.
- Lorenz, E. N. (1984). The local structure of a chaotic attractor in four dimensions. *Physica D: Nonlinear Phenomena*, 13, 90–104.
- Lorenz, E. (1963). Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20, 130–141.
- Lü, J., Chen, G., & Cheng, D. (2004). A new chaotic system and beyond: The generalized lorenz-like system. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, 14, 1507–1537.
- Market Research Future. (2019). Metal Packaging Market Size, Share, Growth | Report, 2030. Date Accessed: 23-05-2022. <https://www.marketresearchfuture.com/reports/metal-packaging-market-1917>
- MarketsandMarkets. (2021). Anti-counterfeit Packaging Market Global Forecast to 2026 | MarketsandMarkets. Date Accessed: 22-05-2022. <https://www.marketsandmarkets.com/Market-Reports/anti-counterfeit-packaging-advanced-technologies-and-global-market-129.html>
- Mordor Intelligence. (2020). Anti Counterfeit Packaging Market Size, Share, Trends (2022 - 27). Date Accessed: 22-05-2022. <https://www.mordorintelligence.com/industry-reports/anti-counterfeit-packaging-market>
- Mortenson, M. E. (1999). *Mathematics for computer graphics applications*. Industrial Press.
- Murphy, A. (1840). *The Works of Samuel Johnson, LL. D. With an Essay on his Life and Genius* (Vol. 1). Alexander V. Blake.
- Nathe, P. D. (2012). Analysis of Security Printing Features Accomplished by Sheetfed Lithographic Offset Process and Sheetfed Screen Printing Process. *I.J.E.M.S.*, 3, 256–259.
- National Electrical Manufacturers Association (NEMA). (2009). Authentication Technologies for Brand Protection.
- O'Connor, J. J., & Robertson, E. F. (n.d.). MacTutor History of Mathematics Archive. Date Accessed: 07/2022 – 04/2024. <https://mathshistory.st-andrews.ac.uk>
- OECD/EUIPO. (2016). Illicit Trade - Trade in Counterfeit and Pirated Goods: Mapping The Economic Impact.

- OECD/EUIPO. (2019). Illicit Trade - Trends in Trade in Counterfeit and Pirated Goods.
- OECD/EUIPO. (2021). Illicit Trade - Global Trade in Fakes: A Worrying Threat.
- Ogundare, J. O. (2018). *Understanding least squares estimation and geomatics data analysis*. John Wiley & Sons.
- Orbis Research. (2019). Global Anti-Counterfeiting Packaging Market Size, Status and Forecast 2019-2025. Date Accessed: 23-05-2022. <https://www.orbisresearch.com/reports/index/global-anti-counterfeiting-packaging-market-size-status-and-forecast-2019-2025>
- Oseledets, V. (1968). A multiplicative ergodic theorem. Characteristic Ljapunov, exponents of dynamical systems. *Tr. Mosk. Mat. Obs.*, 19, 179-210.
- Ott, E. (1993). *Chaos in Dynamical Systems*. Cambridge University Press.
- Pecora, L. M., & Carroll, T. L. (1990). Synchronization in chaotic systems. *Physical Review Letters*, 64, 821-824.
- Pecora, L. M., & Carroll, T. L. (1991). Driving systems with chaotic signals. *Physical Review A*, 44, 2374.
- Perko, L. (2001). *Differential Equations and Dynamical Systems* (3rd ed.). Springer, New York.
- Pikovsky, A., Rosenblum, M., & Kurths, J. (2001). *Synchronization : A Universal Concept in Nonlinear Sciences*. Cambridge University Press.
- Ramirez, J. P., Garcia, E., & Alvarez, J. (2020). Master-slave synchronization via dynamic control. *Communications in Nonlinear Science and Numerical Simulation*, 80, 104977.
- Reinhard, E., Ward, G., Pattanaik, S., & Debevec, P. (2006). *High dynamic range imaging : acquisition, display, and image-based lighting*. Morgan Kaufmann.
- Reports & Data. (2019). Anti-Counterfeit Packaging Market Size - Industry Report, 2019-2026 | Reports and Data. Date Accessed: 23-05-2022. <https://www.reportsanddata.com/report-detail/anti-counterfeit-packaging-market>
- Research Nester. (2022). Metal Packaging Market Size : Global Industry Demand, Growth, Share & Forecast 2024. Date Accessed: 23-05-2022. <https://www.researchnester.com/reports/metal-packaging-market/511>

- reticle - OED. (2023). reticle, n. meanings, etymology and more. Oxford English Dictionary. Date Accessed: 14-08-2023. <https://doi.org/10.1093/OED/4658374409>
- Rössler, O. E. (1976). An equation for continuous chaos. *Physics Letters A*, 57, 397–398.
- Ruelle, D., & Takens, F. (1971). On the nature of turbulence. *Communications in Mathematical Physics*, 20, 167–192.
- Salle, J. P. L., & Lefschetz, S. (1961). *Stability by Liapunov's direct method : with applications*. Academic Press.
- Shi, J., & Tomasi, C. (1994). Good features to track. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 593–600.
- Skokos, C. (2010). The Lyapunov Characteristic Exponents and Their Computation. *Lecture Notes in Physics*, 790, 63–135.
- Smale, S. (1967). Differentiable dynamical systems. *Bulletin of the American Mathematical Society*, 73, 747–817.
- Smithers. (2019). Anti-Counterfeiting & Security Packaging to 2024 | Market Reports | Smithers. Date Accessed: 22-05-2022. <https://www.smithers.com/services/market-reports/packaging/anti-counterfeiting-and-security-packaging-to-2024>
- Sparrow, C. (1982). *The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors* (Vol. 41). Springer New York.
- Spink, J. (2012). Overview of the Selection of Strategic Authentication and Tracing Programmes. In A. I. Wertheimer & P. G. Wang (Eds.). ILM Publications.
- Sprott, J. C. (2003). *Chaos and time-series analysis*. Oxford University Press.
- Sprott, J. C. (2010, January). *Elegant chaos: Algebraically simple chaotic flows*. World Scientific Publishing Co.
- Strogatz, S. H. (1994). *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Perseus Books Publishing, L.L.C.
- Tabor, M. (1989). *Chaos and Integrability in Nonlinear Dynamics : An Introduction*. Wiley.
- van Renesse, R. L. (1998). Verifying versus falsifying bank notes (R. L. van Renesse, Ed.). *Optical Security and Counterfeit Deterrence Techniques II*, 3314, 71–85.

- Verified Market Research. (2018). Authentication and Brand Protection Market Size, Share & Forecast. Date Accessed: 02-04-2024. <https://www.verifiedmarketresearch.com/product/global-authentication-and-brand-protection-market-size-and-forecast-to-2025/>
- Wiggins, S. (2003). *Introduction to Applied Nonlinear Dynamical Systems and Chaos* (2nd ed.). Springer-Verlag.
- Wikipedia Contributors. (2023a). 2008 Chinese milk scandal. Date Accessed: 14-11-2023. https://en.wikipedia.org/w/index.php?title=2008_Chinese_milk_scandal&oldid=1188478669
- Wikipedia Contributors. (2023b). Camera phone. Date Accessed: 14-11-2023. https://en.wikipedia.org/w/index.php?title=Camera_phone&oldid=1196698348
- Wolf, P. R., Dewitt, B. A., & Wilkinson, B. E. (2014). *Elements of Photogrammetry with Applications in Geographic Information Systems* (4th ed.). McGraw-Hill Education.